A Novel Process Model-driven Approach to Comparing Educational Courses using Ontology Alignment

PhD Thesis

Elena A. Chernikova

Software Technology Research Laboratory
Faculty of Technology
De Montfort University
England

Supervisors:

Dr. Francois Siewe
Prof. Georgiy I. Revunkov
Prof. Hussein Zedan

This thesis is submitted in partial fulfillment of the requirements for the Doctor of Philosophy.

May 2, 2014

Declaration

I declare that the work described in this thesis was originally carried out by me during the period of registration for the degree of Doctor of Philosophy at De Montfort University, the United Kingdom, from October 2007 to October 2013. It is submitted as partial fulfilment for the degree of Doctor of Philosophy at De Montfort University. Neither this material nor its part has been submitted for the award of any other degree or qualification to any other higher educational body.

Acknowledgement

It is hard to evaluate, how many people made this research possible by their good will, wise and timely advice and kind support. I am afraid these lines will never be able to express all my gratefulness to them.

I am grateful to De Montfort University and **Prof. Jeffrey Knight**, who made this research possible.

I would like to thank my supervisory team for being wise, thoughtful and inspiring. My deepest gratitude goes to my supervisor **Dr. Francois Siewe**. He was always ready to help and give a piece of good advice. Dr. Siewe is a very patient, supportive and attentive supervisor. Whenever I needed his assistance, he was always there. I am nonetheless thankful to my technical advisor **Prof. Hussein Zedan**. He was always ready to discuss my project and answer numerous questions. Very often his encouragement gave me strengths not to surrender. He is an outstanding person, who created a strong technical community, consisting of incredible people and professionals. I must admit I always felt at home in the Laboratory. My special thanks go to my supervisor in Russia, **Prof. Georgiy I. Revunkov** from Bauman Moscow State Technical University. I highly appreciate his attention and readiness to stay extra hours at the University when I needed his help. I am very grateful to all the academic staff and students of the Software Technology Research Laboratory, including **Dr. Antonio Cau**, **Dr. Helge Janicke** and others. I believe that their sharp questions and invaluable comments given at my seminars at STRL helped me to improve the quality of my work.

I would like to thank my dear parents **Elena** and **Alexander Chernikovs** for their immense love, constant support and encouragement. I could not have coped with the work without their help and understanding. They did their best to make me

feel comfortable and able to carry out this research in many uneasy situations. I am very thankful to my grandfather **Constantine Moukhin**, who contributed enormously to my upbringing. He was very excited about all my undertakings, including the PhD studies, but did not live up to these days. My modest hope is that all my grandparents would have been proud of me.

I am very grateful to my boyfriend **Pavel Nikolaev** for having gone all this long and complicated way together, sharing the joys and sorrows. I am thankful to him for he never stayed indifferent to my research, having been ready to listen and advise. I am also thankful to his family: parents **Svetlana** and **Eugene Nikolaevs**, and grandmother, **Prof. Irina P. Nikolaeva**. I highly evaluate their kindness and support.

I am especially thankful to my employers: the IT company "FB Consult" (2008-2012) and JSC VTB Bank (2012 till present) for understanding, how important my studies are for me. I have learnt a lot at both jobs and these knowledge and skills were of great help in my research. I am very grateful to my chiefs **Kirill Bykov** (FB Consult) and **Maxim Mishin** (VTB Bank) for being extremely supportive and having organized my schedule, so that I managed to combine my work and research. I am also thankful to all my friends and colleagues for their encouragement.

At the very edge of these studies, I must admit that they significantly contributed to my self-development in professional and personal areas. I learnt to overcome myself and do the things, which I thought I would never be able to do. Doing the research made it possible to visit other countries, share opinions and make a lot of friends from all around the world, which is already invaluable.

Abstract

Nowadays, the comparison of educational courses and modules is performed manually by experts in the field of education. The main objective of this research work is to create an approach for the automation of this process. The main contribution of this work is a novel, ontology alignment-based methodology for the automated comparison of academic courses and modules, belonging to the cognitive learning domain. The results of this work are appropriate for such tasks as prior learning and degree recognition, the introduction of joint educational programmes and quality assurance in higher education institutions.

The set-theoretical models of an educational course, its modules, learning outcomes and keywords were created and converted to the ontology. The choice of the information to be presented in the ontology was based on the careful analysis of programme specifications, module templates and Bologna recommendations for the comparison of educational courses. Ontology was chosen as the data model due to its ability to formally specify semantics, to represent taxonomies and to make inferences regarding data.

The formal grammars of a keyword and a learning outcome were created to enable the semi-automated population of the ontology from the module templates. The corresponding annotators were designed in the General Architecture for Text Engineering 6.1.

The algorithm for the comparison of educational courses and modules was based on the alignment of ontologies of their keywords and learning outcomes. A novel measure for calculating the similarity between the action verbs in the learning outcomes was introduced and was utilised. Both the measure and the algorithm were implemented in Java.

For evaluation purposes, we utilised the module templates from the De Montfort and the Bauman Moscow State Technical Universities. The automatically produced annotations of the keywords and the learning outcomes were evaluated against a manually created gold standard. The high values of the precision, recall and f-measure proved their quality and their suitability for the task. The results produced by the alignment algorithm were compared with those produced by human judgement. The results returned by the experts and the algorithm were comparable, thus showing that the proposed approach is applicable for the partial automation of the comparison of educational modules and courses.

Publications

E. Chernikova and P. Nikolaev, "The Similarity Measure and Algorithm for Comparison of the Learning Outcomes", in *Proceedings of the Fourth International Conference on Internet Technologies and Applications*, 2011, pp. 465-473.

Черникова Е.А., Николаев П.Е. Сравнение учебных планов на основе выравнивания онтологий // Сборник тезисов докладов общеуниверситетской научно-технической конференции "Студенческая научная весна - 2011", посвященной 50-летию вылета Ю.А. Гагарина в космос. Том XI, часть 2. - М.:Издательство МГТУ им.Н.Э. Баумана, 2011. - С. 102-104. [E. Chernikova and P. Nikolaev, "The Comparison of Educational Courses based on Ontology Alignment", in *Proceedings of the University scientific and technical conference "The Students Spring - 2011"*, 2011, pp. 102-104.]

Черникова Е.А., Черников А.С. Формализация и сравнение учебных программ на основе онтологического подхода // Вестник МГТУ им. Н.Э. Баумана. Сер. "Приборостроение". Спецвыпуск "Информационные технологии и компьютерные системы", 2011. - С.101-104. [E. Chernikova, A. Chernikov, "Formalisation and Comparison of Educational Programmes based on the Ontological Approach", *Vestnik of BMSTU*, 2011, pp. 101-104.]

E. Chernikova, H. Zedan and F. Siewe, "Ontology-based Semi-Automated Methodology for Module Templates Representation", in "Proceedings of the 2nd International Conference of Creativity and Innovation in Software Engineering", 2009.

Черникова Е.А. Перспективы применения онтологического подхода к описанию учебного процесса в ВУЗе // Проблемы построения и эксплуатации систем обработки информации и управления. Сб. статей Вып.7/под ред. В.М. Черненького. - М.: Издательство МГТУ им.Н.Э. Баумана, 2008. - С. 11-19. [E. Chernikova,

"Prospects of Application of the Ontological Approach to the Description of the Educational Process at University", *Problems of Design and Operation of Information Control and Processing Systems*, no. 7, 2008, pp. 11-19.]

P. Nikolaev and E. Chernikova, "The Ontology Application for Creation and Optimization of Information Management Systems for the International Programme Participating Institutes", in *Proceedings of High-Tech in Chemical Education (Sharing Best Practices of Leading European and Russian Universities)*, 2007, pp. 38-39.

Contents

1 Introduction **1**

1.1 Background 1

1.2 Motivation and Problem Statement 2

1.3 The Research Questions 3

1.4 The Research Methodology 4

1.5 Contributions of the Research 7

1.6 Thesis Outline 8

2 Literature Review **10**

2.1 Introduction 10

2.2 Information Sources for Comparison of Educational Courses and Modules 11

2.2.1 The Internationalisation of Higher Education in Europe . . . 11

2.2.2 The Cornerstones of the Bologna Process 14

2.2.3 Taxonomies of Educational Learning Objectives and Outcomes 19

2.2.4 State of the Art in Educational Course Design and the Comparison thereof 25

2.2.5 Documenting Courses in the UK and the RF 27

2.2.6 Comparison of Competence Models in British and Russian Educational Systems 33

2.3 Ontologies, Similarity Measures and Ontology Alignment 37

2.3.1 Knowledge Representation Models 37

2.3.2 Informal Definition of Ontology 40

2.3.3 Formal Definition of Ontology 42

2.3.4 Description Logics 43

2.3.5 Ontology Languages and DL 44

2.3.6 Semantic Web Rule Language 47

2.3.7 Knowledge Representation System based on DL 50

2.3.8 Ontology Reasoning 51

2.3.9 Ontology Engineering Tool Protégé 52

2.3.10 Ontology Alignment 52

2.3.11 Similarity Measures 53

2.3.12 The General Alignment Process 62

2.3.13 The Ontology Alignment Evaluation Initiative 64

2.3.14 Existing Ontology Alignment Approaches and Algorithms . 65

2.4 Summary 69

3 Ontology of an Educational Course and Modules 72

3.1 Introduction 72

3.2 Methodology for Creation of an Ontology of an Educational Course and Module 73

3.2.1 Ontology Building 73

3.2.2 Ontology Population 74

3.3 The Basic Ontology of an Educational Course and Module 74

3.3.1 The Formal Model Of an Educational Course and Module . 74

3.3.2 Creation of the Ontology based on the Formal Model of an Educational Course and Module 78

3.4 Grammars of the Keywords and the Learning Outcomes 80

3.4.1 Characteristics of the Grammars 81

3.4.2 The Structure of the Grammars 81

3.4.3 The Terminal and Non Terminal Symbols Shared by the Grammars 83

3.4.4 The Rules of the Grammar of the Keywords 83

3.4.5 The Rules of the Grammar of the LOs 86

3.5 The Data Domain Ontology 99

3.5.1 The Formal Model of the Data Domain 99

3.5.2 Creation of the Ontology of the Data Domain based on the Formal Model 100

3.6 Ontology of a Learning Outcome 101

3.6.1 The Formal Model of a Learning Outcome 101

3.6.2 Creation of the Ontology of a Learning Outcome 103

3.7 Combination of the Ontologies 104

3.8 Summary 105

4 Alignment Algorithm for the Ontologies of an Educational Course and Modules **107**

4.1 Introduction 107

4.2 The Alignment Methodology 108

4.2.1 Choosing the Best Matches 110

4.2.2 The Similarity Measure for Comparison of the Words 113

4.3 The Alignment of the Keywords 116

4.3.1 Comparison of the Pairs of Keywords 116

4.3.2 Comparison of the Keywords through the Topics 125

4.3.3 The Total Similarity for a Pair of Keywords 127

4.3.4 Aggregation of the Similarities between the Keywords . . . 127

4.4 Similarity Measure for Comparison of the Action Verbs 127

4.4.1 Adjustments to D. Krathwohl's Taxonomy 128

4.4.2 Semantic Similarity Measure CAVe 129

4.4.3 Sub-Ontology of Verbs 132

4.4.4 Combination of the Ontologies 134

4.5 Comparison of the Learning Outcomes 134

4.5.1 Comparison of a Pair of Learning Outcomes 138

4.5.2 Aggregation of the Similarities for the Learning Outcomes . 143

4.6 Summary 143

5 Implementation **146**
5.1 Introduction . . . 146
5.2 The Process Model of the System . . . 146
5.3 The Recogniser and Ontology Population . . . 148
5.3.1 Implementation of the Recognisers in GATE . . . 149
5.3.2 Recognition of the Keywords . . . 152
5.3.3 Recognition of the Learning Outcomes . . . 153
5.3.4 Population of the Common Classes of the C&M Ontology . 154
5.3.5 Population of the Data Domain Sub-ontology . . . 154
5.3.6 Population of the Learning Outcomes Sub-ontology . . . 155
5.4 Reasoning . . . 157
5.5 Implementation of the Ontology Alignment Algorithm . . . 157
5.5.1 The Loader . . . 159
5.5.2 Data Domain Sub-ontologies Matcher . . . 163
5.5.3 Learning Outcomes' Sub-ontologies Matcher . . . 164
5.5.4 The Aggregator and the Output . . . 165
5.6 Summary . . . 166

6 Evaluation **167**
6.1 Introduction . . . 167
6.2 Description of the Data Sets . . . 168
6.3 Evaluation of the Recogniser . . . 168
6.3.1 Evaluation of the Annotations for the Keywords . . . 169
6.3.2 Evaluation of the Annotations for the Learning Outcomes . 170
6.4 Evaluation of the Alignment Algorithm . . . 172
6.4.1 Evaluation Methodology . . . 172
6.4.2 Experiment using Pairs of Learning Outcomes . . . 178
6.4.3 Experiments with the Modules . . . 182
6.5 Critical Discussion of the Results . . . 190
6.5.1 Critical Discussion of the Annotators' Results . . . 191
6.5.2 Critical Discussion of the Algorithm's Results . . . 192
6.6 Summary . . . 195

7 Conclusions and Future Work **198**

7.1 Research Summary 198

7.2 Revisiting the Research Questions and Contributions 200

7.3 The Scope of Applicability and Limitations 203

7.4 Revisiting the Research Methodology 204

7.5 Future Work 206

7.6 Concluding Remarks 207

Appendix A The Taxonomies of Educational Objectives **220**

Appendix B The Relations of the Ontology of an Educational Course and Module **224**

Appendix C The Relations of the Data Domain Sub-Ontology **229**

Appendix D The Relations of the Sub-Ontology of the Learning Outcomes **231**

Appendix E The Parts of Speech and Other Symbols **233**

Appendix F The Examples of Short Course and Module Templates **236**

Appendix G The Phases of the Keywords' and Learning Outcomes' Annotators **238**

Appendix H The Illustration of the Implementation of the Algorithms **245**

Appendix I The Questionnaire **249**

List of Figures

2.1 The Hierarchy of Competencies in the British Higher Education System 34

2.2 The Hierarchy of Competencies in the Russian Higher Education System 36

2.3 Architecture of a Knowledge Representation System based on Description Logic SROIQ 50

3.1 The Types of Modules' Inclusion in a Programme Specification . . . 76

3.2 The Basic Ontology of a Generic Educational Course and Module . 79

3.3 The Ontology of the Data Domain 101

3.4 The Ontology of a Learning Outcome 103

3.5 Combination of the Basic Course and Module, Learning Outcome and Data Domain Ontologies 105

4.1 The Algorithm for Comparison of the Educational Modules (Courses) based on Ontology Alignment of Keywords and Learning Outcomes 109

4.2 The Greedy Iterative Strategy 111

4.3 The Maximum Average Strategy 114

4.4 The Alignment of the Keywords of Modules (Courses) 117

4.5 Comparison of a Pair of Keywords 118

4.6 Comparison of the Keywords through the Topics 126

4.7 The Ontology of Verbs 135

4.8 Combination of the Course, Module, Learning Outcome, Data Domain and Verbs' Ontologies 135

4.9 The Alignment of the Learning Outcomes of Modules (Courses) . . 136

4.10 Comparison of a Pair of Learning Outcomes 137

4.11 Comparison of the Specifying Clauses 141

5.1 The Process Model of the Semi-automated System for Comparison of Educational Modules and Courses 147

5.2 The Recogniser for a Short Module Template 149

5.3 The Aligner for a Pair of Short Module (Course) Templates 158

5.4 The Loader of a Keyword 160

5.5 The Loader of a Learning Outcome 162

H.1 The Implementation of the Structural and Semantic Similarity between Keywords 246

H.2 The Implementation of the Relation Similarity between Keywords . 247

H.3 The Implementation of Learning Outcomes' Alignment 248

List of Tables

4.1 Comparison of the Keywords' Constituents 119

4.2 Comparison of the Relations' Ranges 121

4.3 Comparison of the Subrelations' Ranges 122

4.4 Comparison of the Verbal Subrelations and Ranges 124

4.5 Comparison of the Keywords of Topics 126

4.6 Two-dimensional Representation of the Cognitive Space of the Revised Taxonomy of Educational Aims 129

4.7 The Minimal Distances between Subcategories of Neighbouring Categories 132

4.8 The Maximal Distances between Subcategories of Each Category . 132

4.9 Comparison of the Data Domain Objects 139

4.10 Comparison of the Specifying Clauses 142

4.11 Comparison of the Learning Outcomes 143

6.1 Evaluation of the Annotations (Keywords) 170

6.2 Evaluation of the Annotations (Relations) 171

6.3 Evaluation of the Annotations (Learning Outcomes) 171

6.4 Evaluation of the Annotations (Learning Outcomes' Constituents) . 172

6.5 Assignment of the Intervals of the Algorithm's Similarity Values to the Answers on the Likert Scale 173

6.6 Comparison of the Pairs of Learning Outcomes 178

6.7 Comparison of the Pairs of Learning Outcomes 179

6.8 Comparison of the Module Templates. Experiment 1 182

6.9 Comparison of the Module Templates. Experiment 1 183

6.10 Comparison of the Module Templates. Experiment 2 185

6.11 Comparison of the Module Templates. Experiment 2 185

6.12 Comparison of the Module Templates. Experiment 3 187

6.13 Comparison of the Module Templates. Experiment 3 187

6.14 Comparison of the Module Templates. Experiment 4 188

6.15 Comparison of the Module Templates. Experiment 4 189

6.16 Comparison of the Module Templates. Experiment 5 189

6.17 Comparison of the Module Templates. Experiment 5 190

G.1 The Correspondence between the Groups of Hepple tags and the Non-terminals in the GK and GLO 239

List of Abbreviations

DL Description Logics

ECTS European Credit Transfer System

EHEA European Higher Education Area

ENIC European Network of Information Centres

GATE General Architecture for Text Engineering

ISCED International Standard Classification of Education

JAPE Java Annotation Patterns Engine

HTML HyperText Markup Language

LO Learning Outcome

MT Module Template

NARIC National Academic Recognition Information Centre

OAEI Ontology Alignment Evaluation Initiative

OWL Web Ontology Language

OWL-DL Web Ontology Language-Description Logics

PS Programme Specification

SMT Short Module Template

SWRL Semantic Web Rule Language

XML EXtensible Markup Language

Mathematical Notation Index

$\in$ - set membership.

$\forall$ - universal quantifier.

$\exists$ - existential quantifier.

$\bigcap$ - intersection.

$\wedge$ - conjunction.

$\rightarrow$ - implication.

$\leq$ - less or equal.

$\geq$ - greater or equal.

Chapter 1

Introduction

1.1 Background

Information systems for universities and education are currently in great demand. Some of these new methods help to improve student-teacher interaction, while others contribute to the worldwide dissemination of knowledge, making it available to as many people as possible. For example, Blackboard facilitates online collaboration between students and academic staff, the storage of and easy access to the learning and teaching materials, the quick spread of education-related news and many other features. Free online lectures conducted by, for example, Stanford University (http://online.stanford.edu/courses), contribute to the worldwide spread of knowledge.

The movement towards the internationalisation of higher education not only takes place on the World Wide Web, but is also supported by official bodies in many countries. However, at this level, the task becomes more complicated than the publishing of separate learning materials. The creation of the European Higher Education Area motivates the advent of joint educational programmes between universities in different countries and encourages academic mobility. The main aim of this process is to create bridges between the educational systems of different countries in such a way that an individual who graduates from one university can make a seamless transition to another, thereby being able to obtain a good position in industry anywhere in the world.

One of the complexities of this task lies in the fact that, on one hand, there is a strong need to make the educational systems and therefore the courses comparable, but, on the other hand, it is also necessary to preserve the unique characteristics of each of them. For this reason, special workgroups were created to identify the criteria and ways of alignment.

The matching of the international educational systems implies the comparison of the educational courses and modules offered by various higher education providers. It is common knowledge that a higher education course is a complicated structure, which demands serious planning and analysis by a team of educationalists and data domain experts. A comparison of the courses and modules is not easier, and demands a certain amount of effort and expertise nonetheless.

Current computer science offers quite an extensive range of methods and means for solving various research problems. For example, the semantic web technologies are a fruitful source of approaches to the solutions of the problems related to data representation and similarity computation.

Ontology is one of these data models and is currently used for creation of knowledge bases, efficient searching of the databases and the Web, semantic searches for information and the storage of a shared, formal description of a data domain [16]. As applied to the higher education information systems, they are utilised to formally describe academic disciplines [62, 81], to store learning materials online [26] and to develop intelligent tools for teaching and for the testing of students [62, 64, 106]. The ontology matching and alignment direction has developed significantly during recent decades. This technique allows the automatic comparison of the formalised data domain descriptions once they are stored in ontologies. Ontology alignment is strongly related to similarity measures theory. The matching algorithms utilise existent and propose new similarity measures, as well as methods for the aggregation thereof.

1.2 Motivation and Problem Statement

The creation of joint educational courses is one of the main directions in EHEA development, as it enhances collaboration between international universities and facilitates academic mobility.

Once two or more universities decide to organise a joint educational programme, they have to design a joint programme specification. To do this, they have to compare the existent educational courses and modules manually. This work demands much effort on the part of highly qualified experts.

This research work is focused on the creation of a novel methodology for the partial automation of the comparison of universities' courses and modules, using a formal, ontology-based approach. The information for comparison is to be extracted from the programme specifications and the module templates, which contain the definitive publicly available information regarding the aims, the intended learning outcomes and the expected learner achievements. In this work, we concentrate on educational courses and modules that belong to the cognitive learning domain. The term "cognitive learning domain" is used in the sense adopted from [20] and covers the area of "stored knowledge and the learnt processes to handle that knowledge" [49, p. 205]. We do not address subjects in the affective or in the psychomotor domains, as these cover areas of learning "related to emotions, systems of values, motivation and (social) behaviour" [49, p. 205] and "where physical movement andor coordination are involved" [49, p. 205], respectively.

The main objectives of the research are as follows:

1. To create an ontology for the formal representation of an educational course, its modules, learning outcomes and keywords.
2. To propose a methodology for the partially automated population of the ontology based on programme specifications and module templates.
3. To design, implement and evaluate an ontology alignment algorithm for ontologies of the educational courses and modules.

1.3 The Research Questions

The research work aims to answer the following question.

How to automate the comparison of higher education courses and modules, belonging to cognitive learning domain, using documents and an ontology-based approach?

The main research question and the objectives of this work are the following. Each of them is related to the objective that has the same number.

1. Which information about educational courses and modules should be used for comparison and how will it be stored in an ontology?
2. How to automate the population of the ontology with the data from the documents?
3. What is the alignment algorithm for ontologies of educational courses and modules? Which similarity measures should it utilise?

1.4 The Research Methodology

The choice of research methodology depends strongly on the nature of the research question [93]. We aim to automate the comparison of educational courses and modules. This problem touches on such fields of knowledge as computer science, software engineering and education theory. In order to solve it, we wish to create a process model, which would produce the same results in the comparison of educational courses and modules as if the comparison had been performed by a human expert. We do not take into account the cultural aspects of the process. Our work is based on the Bologna Workgroup's recommendations, which address the countries of the European Higher Education Area.

We find constructive research methodology suitable and appropriate in our case. Neither qualitative, nor quantitative approaches, which are widely used in social research, are sufficiently exhaustive for our task. This is because they mainly consider data collection and analysis, such as case studies, questionnaires and surveys, or concentrate on improving existent solutions like experiments, while our aim is to construct a novel methodology based on the results of the analysis. At the same time, constructive research, according to [38], "implies building of an artifact (practical theoretical or both) that solves a domain specific problem in order to create knowledge about how the problem can be solved (or understood, explained or modelled) in principle". Thus, constructive research methodology satisfies our demands. We also utilise the research methods of qualitative and

quantitative approaches in the literature review and evaluation stages of the research process.

According to [78], constructive methodology contains the following steps.

1. Find a practically relevant problem that also has research potential.
2. Obtain a general and comprehensive understanding of the topic.
3. Innovate an idea for a solution and develop a problem-solving construction, which also has the potential for theoretical contribution.
4. Implement the solution and demonstrate that it works.
5. Show the theoretical connections and the research contribution of the concept solution.
6. Examine the scope of applicability of the solution.

The first stage is addressed in Sections 1.1 and 1.2 of the current chapter. The automation of the educational courses and the comparison of the modules is an on-going problem with research potential in the domains of both education and computer science.

The second stage is presented in Chapter 2, which contains a literature review. The first part of the review studies current approaches to the comparison of educational courses and modules. We performed document-based research in order to define the scope of information necessary for comparison. We did not use methods such as surveys or questionnaires, because this would have been a time consuming process. We also assumed that the Bologna documents constitute the agreed positions of international educationalists. As a result, utmost importance was given to the module's keywords and learning outcomes. The second part of the review is devoted to ontologies, ontology alignment algorithms and similarity measures. The research method involved an analysis of relevant documents, including journals, conference papers and books.

The third stage is discussed in Chapters 3 and 4. The third chapter presents the ontology of an educational course and it constituents. We utilised the modelling research method to create an ontology based on set-theoretic models of an educational course, module, its keywords and its learning outcomes. We also created

formal grammars for a keyword and for a learning outcome. The fourth chapter contains a description of the algorithm for alignment of the ontologies of the courses and the modules based on the keywords and learning outcomes. The theoretical computer science research method was applied to the design of this algorithm.

The fourth stage is presented in Chapters 5 and 6. It covers the implementation and evaluation of the approach. The fifth chapter describes the implementation of the system with the help of object-oriented design methodology.

The sixth chapter contains an evaluation of the annotators of keywords and learning outcomes, as well as of the ontology alignment algorithm. We combined qualitative and quantitative data analysis methods for this evaluation.

The results produced by the keywords' and learning outcomes' annotators were compared to the manually created gold standard. The quantities of recall, precision and f-measure were computed. We chose this method because it is commonly used to evaluate the performance of annotators.

Comparison of the results produced by the algorithm versus human judgement was chosen as an evaluation methodology. This approach was utilised due to the nature of the research task, which included semantic similarity computation and which aimed to automatically produce the same results as the human experts. Moreover, to the best of our knowledge, none of the current systems provides alike functionality, as ours does. For this reason, we could not compare the results to an analogue. In order to evaluate the ontology alignment algorithm, we created five case studies and a questionnaire, in which the experts were asked to evaluate the similarity of the pairs of short module templates according to the Likert scale. The results were analysed by means of statistical methods.

The fifth stage is touched on in the following chapters of the thesis. Chapter 2 provides the theoretical basis for this research and, at the same time, shows the connections to education science (considering the comparison of educational courses and modules) and to the semantic web (considering ontology alignment). The sections 1.5 and 7.2 outline the contributions of this research. The entire chapter 7 is devoted to a discussion of the results of the work conducted.

The sixth stage of the methodology is discussed in the current chapter and in chapter 7. Section 7.3 outlines the scope and limitations of the designed methodology.

1.5 Contributions of the Research

The main contribution of this work is a novel, ontology alignment-based methodology for the automated comparison of academic courses and modules belonging to cognitive learning domain.

The designed methodology yields the following minor contributions as its constituents.

1. The novel ontology of the course, module, keywords and learning outcomes.
2. The novel formal grammars and annotators for the keywords and the learning outcomes.
3. The novel similarity measure CAVe for comparison of the action verbs of the learning outcomes in the cognitive domain of D. Krathwohl's taxonomy of educational aims.
4. The novel designed, implemented and evaluated ontology alignment algorithm for the comparison of the educational modules and courses.

The first contribution answers the first research question. In our research, it solves the problem of data representation in the system. At the same time, this ontology is an independent contribution to knowledge, because it can be reused for the formal representation of educational courses in other projects.

In accordance with the Bologna recommendations and the analysis of the supporting documentation for the educational courses, it was decided to base the comparison of educational courses on a calculation of the similarity between the learning outcomes and the keywords of the modules. Ontology was chosen as the data model due to its ability to specify the semantics, to represent the taxonomies and to make formal inferences regarding the data in a precise manner. A set-theoretic model of the courses and modules, their keywords and learning outcomes was created and was then converted to an ontology.

The second contribution answers the second research question. The approach to the semi-automated population of the ontologies from the module templates was proposed. We automated the enrichment of the keywords and the learning

outcomes sections of the ontologies. In order to do this, we designed the formal grammars of a keyword and a learning outcome, based on studies of the "Keywords" and "Learning outcomes" sections of the module templates from the De Montfort University (Leicester, the UK) and Bauman Moscow State Technical University (Moscow, Russia). The grammars were implemented to automate the annotating of the corresponding sections of the module templates with the titles of the ontology's entities.

The third and fourth contributions answer the third research question.

The alignment algorithm for the ontologies of courses and modules is based on a comparison of the modules' keywords and learning outcomes.

A novel similarity measure for the action verbs of the learning outcomes was introduced, based on a revised version of B. Bloom's taxonomy of educational aims. Each action verb is clustered in two-dimensional space, whereafter the similarity between the LOs is measured based on the distance between them inside the framework.

1.6 Thesis Outline

The thesis has the following structure.

Chapter 2: Background and Literature Review

This chapter contains the study of the current situation in two fields, namely the comparison of educational courses and modules, and ontology alignment. Thus, it consists of the following two main sections.

"Information Sources for the Comparison of Educational Courses and Modules".

The first section reviews the design and comparison of courses and modules in universities. The main outcome of this section is the statement that educational courses should be compared based on the keywords and learning outcomes of the modules.

"Ontologies, Similarity Measures and Ontology Alignment".

The second section contains a critical review that covers ontology design and alignment, applications of ontologies in software systems, the basic concepts

of Description Logics and their use in ontology representation languages, and the major algorithms and software tools for ontology engineering and alignment.

Chapter 3: "Ontology of an Educational Course and Modules".

This chapter depicts the ontology of educational modules and courses and proposes the grammars of keywords and learning outcomes, which allow for the semi-automated population of them from the programme specifications and module templates.

Chapter 4: "Alignment Algorithm for the Ontologies of Educational Courses and Modules".

This chapter describes the alignment algorithm for ontologies of educational modules and courses. It introduces a new similarity measure, CAVe, to compare the ontologies of the learning outcomes.

Chapter 5: "Implementation".

This chapter describes the implementation of the keywords' and learning outcomes' annotators based on the corresponding grammars. The chapter also covers the implementation of the similarity measure CAVe and the ontology alignment algorithm in Java. It contains the functional scheme of the entire system.

Chapter 6: "Evaluation".

This chapter provides a detailed description of the experiments conducted and the evaluation of the annotators and the ontology alignment algorithm based on the module templates from the De Montfort University and the Bauman Moscow State Technical University. The annotators are evaluated using the AnnotationDiff tool provided by GATE. The algorithm's performance is compared to the results received from the questionnaires.

Chapter 7: "Conclusions and Future Work".

This chapter summarises the results of the research, discovers its strengths and weak points, and identifies areas that require improvement. Future research goals are set.

Chapter 2

Literature Review

2.1 Introduction

In this chapter, we present the literature review concerning higher education in Europe and ontologies, ontology alignment and how these techniques can be applied to automate the comparison of educational courses and modules. The chapter is divided into two main sections.

In the first section, we describe the current situation in international higher education, particularly in Europe. We wish to first establish that the automation of the comparison of educational courses and modules is an important, up-to-date task. We will then explain the information that we used to partially automate the comparison and will present the reasons for choosing these data and not others.

The section is structured in the following way. Firstly, we depict the legal basis for the internationalisation of higher education. We trace the history of the process from 1997 to the present, touching on the main concepts of the Lisbon, the Sorbonne and the Bologna Declarations. We review the four cornerstones of the Bologna Process, which include the three levels of education, profiles (fields of study), the European Credit Transfer System and the learning outcomes, together with competences. The prevailing situation in and recommendations for the design and comparison of the educational courses and modules are described. As an example, we analyse the ways in which courses and modules are documented in British and Russian universities, and compare the competence models realised in

these countries. As a result of this research, we conclude that educational courses can be treated as sets of the modules of which they consist. We also decided to base the comparison of the educational modules on their keywords and learning outcomes.

To the best of our knowledge, current comparisons of educational courses and modules are performed manually. This procedure is necessary for the creation of joint educational programmes and for recognition of prior learning. In order to facilitate the work of highly qualified experts, we designed a methodology that aims to partially automate the process of comparison of the courses and modules, which will assist the experts in their decision-making processes.

The main objectives of the second section of this chapter are as follows. Firstly, we will choose the formal model for the representation of the data from educational courses and modules. We should then study the model's capabilities, the manner of implementation and further use. Finally, we will explain how the information presented by means of the model can be utilised for the task of comparing the courses and modules.

The second section is structured in accordance with the goals thereof. We reviewed current knowledge representation models and came to the conclusion that ontology was best suited to the stated task. We then focus on ontologies, including the analysis of their various definitions, an overview of description logics, ontology languages and ontology engineering tools. The latter subsection is devoted to ontology alignment and similarity measures. On balance, we concluded that ontology was the most suitable knowledge representation model for educational courses and modules. We decided to design a specialised ontology alignment algorithm for comparison of the corresponding ontologies.

2.2 Information Sources for Comparison of Educational Courses and Modules

2.2.1 The Internationalisation of Higher Education in Europe

Lisbon Recognition Convention (1997)

Collaboration between the European countries has increased across many fields in recent decades, including that of higher education. The signing of the "Con-

vention on the Recognition of Qualifications concerning Higher Education in the European Region" in 1997, also known by its shorter title, the "Lisbon Recognition Convention", was the first step towards the creation of the European Higher Education Area. The main aim of this convention was to facilitate the recognition of higher education degrees awarded in the countries that had already joined or which will join the EHEA. It also declared an individual's right to study wherever s/he wishes, provided his or her prior learning achievements are sufficient to enter a particular educational programme. The convention stands for the creation of a transparent, coherent and accessible educational area worldwide [113].

The Bologna Declaration (1998)

In 1998, the Sorbonne's "Joint Declaration on Harmonisation of the Architecture of the European Higher Education System" followed the EHEA's initiative [2]. It was accepted by the Ministers for Education of the United Kingdom, France, Italy and Germany. It strengthened the understanding of the necessity of enabling people to obtain the best possible education in their field of study by taking advantage of learning opportunities in different countries. It became obvious that, in order to achieve this goal, it was necessary to introduce a framework that would bridge the differences in European HE systems through a set of shared concepts and which would not damage unique educational approaches based on invaluable experience.

The 1999, Bologna Declaration, signed by 29 European countries, and the further development of Bologna process, which now involves 47 states including the Russian Federation and the United Kingdom, contributed to the strengthening of relations between the European higher education providers.

The Bologna Declaration stated six main objectives, the aim of which was to establish the EHEA and to promote the European system of higher education worldwide. The key aspects, as they appear in the variant of 1999, include the following points [3]:

1. Adoption of a system of easily readable and comparable degrees.
2. Adoption of a system essentially based on two main cycles, that of undergraduate and graduate.

3. Establishment of a system of credits as a proper means of promoting the most widespread student mobility, such as in the European Credit Transfer System.

4. Promotion of mobility by overcoming obstacles to the effective exercise of free movement.

5. Promotion of European co-operation in quality assurance with a view to developing comparable criteria and methodologies.

6. Promotion of the necessary European dimensions in higher education, particularly with regard to curricular development, inter-institutional cooperation, mobility schemes and integrated programmes of study, training and research.

The aim of this work is to automate the comparison of the educational courses provided by different universities and to introduce a set of computational models and algorithms for this purpose. This will provide software support for the Bologna process in terms of the latter three points of the Declaration, using the results of the achievement of the first three aims. Comparison of educational courses is related to the main objectives of the Bologna process in the following ways.

Mobility Student mobility is strongly related to the problem of prior learning recognition. Mobility in education means that a learner is able to choose any university in the EHEA, either to start or to continue his or her studies. However, both the student and the educational body must understand whether s/he is eligible to start or continue learning with regard to his or her prior education. For this reason, the university needs to compare the contents and results of the previous learning with the descriptions of available courses. The results of such a comparison may be used to recognise the student's previous achievements and to avoid repetition in the educational modules learnt.

Quality Assurance Comparison of educational courses and modules may be used for quality assurance, by either one particular university or by several educational bodies. The first possible use is a case in which a HE

provider wants to find similar educational modules inside a single educational course and to eliminate repetition or to group a number of modules into one. Another situation is the comparison of the university's courses to another, "ideal" course. The similarity between an educational programme and a standard may be used as one of the quality criteria to judge whether the proposed course is applicable.

Inter-institutional Cooperation Encouragement of inter-institutional cooperation throughout Europe is one of the significant missions of the Bologna process. The introduction of joint and double educational degrees is one of the main types of such collaboration. In order to do this, the universities' partners need to compare the educational courses and the existing modules. The automation of this process could contribute to the development of joint educational programmes.

2.2.2 The Cornerstones of the Bologna Process

The main objective of the Bologna Process is the creation of the European Higher Education Area, whereby both students and academic staff will have mobility in terms of choosing where they wish to study and to teach, while the higher education institutions will hold joint and double education programmes. Much attention is paid to the increase and maintenance of high quality standards in European higher education. Furthermore, the Bologna process propagates a learner-centred approach to teaching. All these aims presuppose that the educational courses taught by the European universities are comparable. This implies the introduction of a framework in higher education that will be shared by all the participants and will allow them a certain amount of flexibility in educational programme design.

The document, called the "Framework for Qualifications of the EHEA", was signed by the members of the Bologna Working Group on Qualification Frameworks in 2005. This is the most complete report regarding such questions as the higher education qualifications in Europe, the national frameworks in the EHEA and the linking of frameworks for qualifications in HE. The document describes the methods for building a system of easily readable and comparable degrees and gives recommendations on how to achieve this through national higher education

policies. The "Framework for Qualifications of the EHEA" is " an overarching framework that makes transparent the relationship between European national higher education frameworks of qualifications and the qualifications they contain. It is an articulation mechanism between national frameworks " [22, p. 29]. For this reason, it was used as a source of information for the organisation of the European Higher Education Area.

The "Framework for Qualifications of the EHEA" describes the following cornerstones of the Bologna process.

1. The three-level system of higher education, whereby each is characterised by a set of Dublin Descriptors [61].
2. Profiles or fields of learning to which the qualifications belong [22].
3. ECTS credits, which depend on the workload students need in order to achieve expected learning outcomes [41].
4. Learning outcomes and competences, which are at the core of the learner-centred educational process [22].

The Cycles and Levels of Higher Education and the Dublin Descriptors

The Bologna Qualification Framework divides higher education into three main sequential levels, called cycles. These are Bachelor's, Master's and Doctoral studies. The fourth type is called a "short cycle". This term is used to classify programmes of study that are "within the Bologna first cycle, but which do not represent the full extent of this cycle" [61, p.1]. The framework of the cycles facilitates comparability of degrees and thus recognition of qualifications by different countries. Each of the cycles is characterised by a set of Dublin Descriptors, which contain the generic statements of what a person must achieve and be able to demonstrate on completion of an educational course at a particular level.

The three Bologna educational cycles are aligned with the higher education levels presented in the International Standard Classification of Education (ISCED). ISCED belongs to the Family of United Nations International Economic and Social Classifications and is used to organise educational programmes and qualifications according to education levels and fields. In short, Bachelor's, Master's and Doctorate cycles coincide with the fifth to eighth ISCED levels, respectively.

Profile of a Higher Education Qualification

In [22, p.30], a profile, as applied to HE qualifications, is determined as "either the specific (subject) field(s) of learning of a qualification or the broader aggregation of clusters of qualifications or programmes from different fields that share a common emphasis or purpose". In our work, we have used the first definition. In this case, the word "profile" is a synonym for the concept "field of education" or "field of study". ISCED defines the field of education as a "[b]road domain, branch or area of content covered by an educational programme, course or module" [6, p. 87]. This notion is widely known and is used in universities throughout Europe. On completion of studies, a learner normally receives a certain degree, such as a Bachelor's or a Master's, in a particular field.

The question of classifying educational programmes in accordance with profiles is a difficult one, as each country may have its own taxonomy of the fields of study. In [22, p.73], it is mentioned that "professional profile is a matter for national sovereignty". Currently, one European classifier of the fields of education is relatively widely known and is included in the International Standard Classification of Education. However, it is not widely used. ISCED distinguishes twenty-five fields of study, divided into nine large groups. At present, UNESCO is revising this classification. The new version was presented at the 37th session of the UNESCO General Conference in November 2013 [114].

ECTS Credits and Workload

The European Credit Transfer System is a further cornerstone of the Bologna process. It has already been legally adopted and is used in the countries in the European Higher Education Area.

According to [22, p.29, 30], a credit is "a quantified means of expressing the volume of learning based on the achievement of learning outcomes and their associated workloads", while 'workload' is defined as "a quantitative measure of the learning activities that may feasibly be required for the achievement of the learning outcomes". The workload is based not only on the number of contact hours, but also includes time for self-study. A learner receives a certain number of credits for having completed a particular amount of work. The number of credits does not reflect whether the task was executed well or not. A student will receive

the credits as long as s/he performed satisfactorily. In accordance with [41, p. 11], "one credit corresponds to 25 to 30 hours of work".

The main idea behind the use of the ECTS is to make the results achieved by the learner comparable and acceptable in any country of the EHEA. This facilitates transfer of a student from one educational course to another within one educational cycle and makes it possible to continue education at a higher level in another university or even another country, by making the degrees recognisable.

The ECTS facilitates learner-centred education, because it enables the construction of an educational programme of blocks, as each educational component is associated with a certain workload and a number of credits. The blocks may contain a variety of disciplines with the same number of credits. Thus, a student may have a flexible pathway in learning and can achieve the same qualifications by different routes.

In accordance with [41], "a fulltime year of formal learning (academic year)" equals 60 ECTS credits, a short cycle qualification –approximately 120 ECTS credits, first (Bachelor) and second (Master) cycles 90-120 and 180-240 respectively, while a third cycle qualification (doctorate studies) may not have credits at all.

Competences and Learning Outcomes

The notions of a learning outcome and a competence are strongly related [67]. Currently, there are different versions of what they are and how they are connected. To obtain a coherent picture, we adopted the definitions of the Bologna documents.

According to [21, p. 6], competences "represent a combination of attributes (with respect to knowledge and its application, skills, responsibilities and attitudes) and are used to describe the level or extent to which a person is capable of performing them". The Framework for Qualifications of the European Higher Education Area declares that learning outcomes are "statements of what a learner is expected to know, understand and/or be able to do at the end of a period of learning" [22, p.29]. It is emphasised that the concept of a competence is broader than that of a learning outcome. In fact, a learning outcome is a type of competence.

The Dublin Descriptors are generic competences, because they depict the student's achievements at a high level of abstraction and are independent of the field

of study. In accordance with [22, p.65], they "offer generic statements of typical expectations of achievements and abilities associated with qualifications that represent the end of each of a Bologna cycle". The Descriptors characterise the nature of the qualification gained at a certain educational level. Five descriptors characterise each of the educational cycles (short courses, Bachelor's, Master's and Doctoral). They illustrate how competent a student should be on completion of a certain level in one of the following directions: knowledge and understanding, applying knowledge and understanding, making judgements, learning focus and communication. For example, the Dublin Descriptors state that, when applied to the learning focus, students who have finished studies at a particular level should have the learning skills to undertake further studies with some autonomy (short cycle), with a high degree of autonomy (Bachelor's cycle), in a manner that may be largely self-directed or autonomous (Master's cycle) and should be able to promote, within academic and professional contexts, technological, social or cultural advancement in a knowledge-based society (Doctoral cycle).

The competences of an educational course are more precise, but still operate using generic data domain concepts. A template of an educational module contains subject-specific competences, which are the learning outcomes.

In early 2000, the Bologna working group turned its attention to the learning outcomes and their critical role in achieving the Bologna group's objectives. Research was conducted on their application in European national higher education frameworks. In 2004, 97% of the countries that were participants of the Bologna Declaration reported some activity in the field, including initiatives at the institutional or state level [21, p. 19]. This process shows the growing popularity of outcomes-based education and the movement from the teacher-centred to the student-centred approach to learning. It becomes obvious that the description of the contents of an educational module or course is no longer sufficient to give a complete picture of what a student will learn. The important role of the learning outcomes is to link the concepts from the data domain of an educational module or course to the learner through a description of the actions s/he will be able to perform on completion of his or her studies. This approach also helps to strengthen connections between higher education and the real demands of enterprise. It facilitates the correspondence of the graduate's qualifications to the employer's expectations, as the learning outcomes of particular educational programmes can be compared to the competences required for a certain position in industry. For

example, in the UK, the "National Occupational Standards (NOS) describe what an individual needs to do, know and understand in order to carry out a particular job role or function" [5]. This definition shows that the nature of an NOS is the same as that of competences and learning outcomes in higher education.

2.2.3 Taxonomies of Educational Learning Objectives and Outcomes

We studied the works of the educationalists B. Bloom [20] and D. Krathwohl [72], who devoted their research to the problem of the classification and ordering of the educational objectives and the learning outcomes. B. Bloom (1956) created the first taxonomy of educational objectives, while his former student, Krathwohl (2002), presented an improved version. We assumed that these classifications were useful for comparison of the educational courses and modules. We reviewed both works with the aim of creating a novel similarity measure for the learning outcomes based on one of the taxonomies of our choice. This decision was taken based on the reasoning that, when comparing the learning outcomes, an expert is extremely likely to consider not only the common sense of the sentences, but also the type and level of the knowledge, ability or skill that a student will obtain.

Requirements for the Novel Similarity Measure

We imposed the following requirements when designing the similarity measure.

Firstly, similarity measures are usually utilised in automatic data processing. Therefore, the calculations should include only such parameters whose values can be automatically extracted or inferred from the text, or collected from a data (knowledge) base, based on the words in the statements. The less the similarity measure is dependent on the statement's construction, the better. This condition implies that a relevant semantic similarity measure should detect and utilise the most important data in the sentence for its calculation and should ignore noise. Therefore, it should produce high values for statements that express the same things, depending as little as possible on minor details and clarifications that an author could have omitted.

Secondly, the measure should be sensitive enough to produce a variety of values. In other words, it should be able to evaluate semantic similarity at a somewhat rough level.

Finally, a novel measure should preserve the properties of any similarity measure presented in Section 2.3.11.

We studied both variants of the taxonomy with regard to their applicability for the creation of the semantic similarity measure.

Blooms' Original Taxonomy of Educational Objectives (1956)

Benjamin S. Bloom, who was then the Associate Director of the Board of Examinations of the University of Chicago, became a pioneer in this field when he proposed the taxonomy of educational objectives, also known as "original taxonomy", in 1956 [20, 49].

In order to agree on the terminology, it is important to mention that some educationalists distinguish between educational objectives and learning outcomes in the following ways. For example, according to [21], the educational objectives reflect that which a lecturer plans to teach, while the learning outcomes describe that which a student will learn. At the same time, [72, p.212] states that "the taxonomy of educational objectives is a framework for classifying statements of what we expect or intend students to learn as a result of instruction". This indicates that the author did not distinguish between the educational objectives and the learning outcomes in the modern sense. For this reason, we assumed that the taxonomy could be applied to the learning outcomes.

Bloom's original taxonomy is presented in full in the Appendix A.

The main aim of the classification was to divide the indivisible, or to partition the knowledge into categories inside the cognitive domain. Furthermore, the author's task was also to design the beacons that would help a course or a module leader to construct an educational objective or a task for a test in such a way that it could be easily, if not automatically, classified into one of the categories.

As far as the first problem is concerned, the educationalist distinguished six categories, namely Knowledge, Comprehension, Application, Analysis, Synthesis and Evaluation. All of these, with the exception of Application, contain several

subcategories. It is important to note that the categories of the cognitive domain and the subcategories contained in each are arranged in order of increasing complexity. This means that mastery of a simpler category is a prerequisite for the mastery of a more complex one.

Bloom's solution to the second problem lay in the assignment of the sets of the verbs used in the statements of the educational aims and the tests' tasks to the categories and subcategories of his model. This decision enabled the classification of the educational objectives, the learning outcomes and the tasks in the tests based on the verbs that they contain.

The main purposes of this classification were the following. Firstly, it enabled the alignment of the tasks in a test with the educational goals set by the module leader. The model could help to evaluate whether all the goals that were set are checked during the students' assessment. Secondly, it is useful for general understanding of the complexity of an educational unit. Thirdly, the fact that the taxonomy is ordered from simple to complex activities can also enable the evaluation of the quality of the planning of a module or a course, as a student must have finished simpler courses before learning aspects that are more sophisticated.

Revised Taxonomy of Educational Objectives (2001)

In 2001, another American educationalist, Krathwohl, continued the work started by Bloom [10]. He improved on his predecessor's classification and called it the "revised taxonomy". Krathwohl (presented in full in Appendix A) made the following significant changes to Bloom's model.

Krathwohl emphasised that a serious drawback of the original taxonomy was that its Knowledge category "embodied both noun and verb aspects" [72, p. 213]. He also mentioned that an objective contains subject matter content expressed by a noun phrase, and a description of an action in this regard, expressed by a verb phrase. He created a second 'Knowledge dimension' and stated that the noun phrase would enable the classification of the statement it contains, while the verb will still be used to categorise the cognitive process dimension. According to Krathwohl [72, p. 214], the knowledge dimension consists of four major categories. These are Factual Knowledge (including the basic elements of knowledge,

such as terminology, specific details and elements); Conceptual Knowledge, characterising the relationship between the basic elements and concepts (including classifications and categories, principles and generalisations, theories, models and structures); Procedural Knowledge (including methods of enquiry, and criteria for using skills, algorithms, techniques and methods) and Metacognitive Knowledge (including strategic knowledge and knowledge about cognitive tasks).

The second change to Bloom's taxonomy was the reorganisation and partial renaming of the categories and subcategories. Similar to that of his predecessor, Krathwohl's revised taxonomy contains six categories, which are then divided into subcategories. Each category has a verb as its title, while each subcategory is a gerund. Krathwohl mentioned in [72, pp. 214-216], the subcategories are only ordered from simple to complex inside their own category. However, it is important to note that not all subcategories of a lower level category need less mastery than do any subcategories of a higher-level category.

Krathwohl (2002) proposed to present his model as a table, in which the rows represent the knowledge dimension and the columns indicate the cognitive process dimension's categories. The educational goals and outcomes are to be clustered into cells. In the event that a statement contains verbs belonging to several categories, the outcome should be allocated to several cells.

Analysis of the Taxonomies' Applicability for the Creation of the Similarity Measure

It is important to emphasise that Krathwohl's (2002) classification is not a novel or separate work, but is an improved version of Bloom's original taxonomy of educational objectives. This is described in the work of [91] as: B. Bloom's revised taxonomy of educational objectives. This means that we do not compare two different classifications, but original and revised versions of the same thing.

Analysis of the Original Taxonomy

Bloom's original taxonomy meets the criteria stated in Section 2.2.3 in the following ways.

Firstly, all the levels of the taxonomy comprehend the action and object features. This means that a learning outcome can be classified according to a level only if

both the verb defining the action and the noun phrase defining the object satisfy its requirements. On one hand, a large number of action verbs are assigned to each of the levels of the original taxonomy. On the other hand, very few nouns are explicitly related to them. Furthermore, noun phrases may be formulated according to numerous variants not containing the selected words, which would undoubtedly result in their being assigned to a particular level. This aspect severely complicates the process of the automatic extraction of these features from the natural language sentences. Secondly, the taxonomy contains only six major levels. This means that a similarity measure would be able to operate only within these levels, thus returning a limited number of different values. These facts indicate that it might be complex to create a similarity measure based on the original taxonomy.

Analysis of the Revised Taxonomy

We critically reviewed both the Knowledge and the Cognitive Process Dimensions of the revised taxonomy in terms of their ability to satisfy the criteria to the similarity measure described in Section 2.2.3.

Analysis of the Knowledge Dimension

We analysed the statement by Krathwohl (2002) regarding the possibility of defining the position of a learning outcome in the knowledge dimension that was based only on the nouns describing the object in the data domain. We propose a counterexample to this proposition.

For instance, let us consider the following simple learning outcomes.

1. Describe **entity-relationship data model**.
2. Create **entity-relationship data model**.

Both of the learning outcomes contain the same noun phrases as their objects (distinguished in **bold** font), and differ only in the action verbs (describe and create). The phrase contains the word ”model” which, according to [72, p.214], could be the token for referring it to Conceptual knowledge (”knowledge of theories, models, and structures”). Thus, if we consider only these noun phrases, we would classify both outcomes as the Conceptual Knowledge dimension. While this is obviously correct for the first example, it is arguable for the second. The

latter sentence describes a practical action over the object and thus leans more towards Procedural knowledge (”Knowledge of subject-specific techniques and methods”, according to [72, p.214]). However, if keeping strictly to the policy of defining the level in the Knowledge Dimension based on the noun only, we would classify this learning outcome wrongly as Conceptual knowledge. This led to the conclusion that, although the noun phrase is required, it is not always sufficient for the classification of the outcomes in the Knowledge dimension. In addition, we encounter the same obstacles as we did with Bloom's original taxonomy. The noun phrases may not contain the specific words that would enable the classification to any of the levels in the Knowledge dimension. This means that it would be problematic to automatically classify a learning outcome as a Knowledge dimension.

The other drawback of the Knowledge dimension in terms of its applicability for the construction of the similarity measure is that it contains only four coordinates. Consequently, the potential measure will only be able to return an extremely limited number of values.

Considering the above arguments, we decided not to base our semantic similarity measure for the Knowledge Process dimension of Krathwohl's taxonomy.

Analysis of the Cognitive Process Dimension

The cognitive process dimension represents a taxonomy of verbs. It contains six major categories, namely Remember, Understand, Apply, Analyse, Evaluate and Create. Each of them contains several subcategories, as shown in Appendix A. For example, the category Remember contains the subcategories Recognising and Recalling. The verbs that name each of the categories are ordered from simpler to more complicated actions. The taxonomy of the Cognitive Process dimension is very detailed and well organised, and is only concerned with action verbs. It is possible to classify a learning outcome according to one of its categories or subcategories, based only on the action verb and independent of the object of the action.

Many universities, for example the University of Limerick, Glasgow University and Dublin City University, publish guidance for writing learning outcomes [66, 80, 94]. These tutorials contain sections with recommendations on the use of particular action verbs in the learning outcomes. They also assign groups of

verbs to the taxonomy's categories. This means that it is possible to automatically classify a learning outcome according to one of the taxonomy's categories based on the action verb used. Furthermore, it is possible to create a model of the taxonomy and to assign the recommended verbs to each category or subcategory, as well as enriching it with other action verbs if necessary.

The analysis of the Cognitive Process dimension of Krathwohl's (2002) taxonomy showed that it is appropriate for being used as the basis of a semantic similarity measure for the action verbs of the learning outcomes.

2.2.4 State of the Art in Educational Course Design and the Comparison thereof

The proposed technical decision must reflect the approaches used in real life situations. For this reason, the current methods of comparison of educational courses were studied. In order to understand how the comparison of educational courses is carried out, the most recent tendencies in academic recognition procedures as recommended by the Bologna Working Group were reviewed.

The Tuning Project

The project, called Tuning Educational Structures in Europe, was launched in 2000 and was directed by universities in the Netherlands and in Spain. It involves about 130 higher education institutions in Europe. The main objective of the project is to find or to create points of reference and a common understanding in the arena of higher education qualifications. The first phase of the project resulted in identifying the point of reference for competencies in several fields of study.

The Tuning project bases all its findings on the learning outcomes-based approach. It has already created sets of subject-specific competencies for several pilot disciplines and has started to investigate the relationship between learning outcomes and teaching, learning and assessment strategies. This shows the significance of competencies and learning outcomes for the creation of a system of comparable degrees in the European Higher Education Area.

ENIC-NARIC Network

The European Network of Information Centres in Europe (ENIC Network) were established by a decision of the Committee of Ministers of the Council of Europe and the UNESCO Regional Committee for Europe in 1994. In accordance with the Lisbon Recognition Convention, the ENIC Network oversees, promotes and facilitates the implementation of its main objectives. Currently, this organisation includes 49 countries worldwide, each of which hosts bodies set up by national authorities, called National Academic Recognition Information Centres (NARIC). They provide information regarding the recognition of foreign diplomas, educational systems, opportunities for studying abroad and so on. The main function of the NARICs is to support people who want to study or work abroad and who need to have their prior education recognised for this purpose. NARICs either provide this service themselves (for example, in the UK) or redirect the applicants to a responsible authority in the country in question (for example, in Russia).

The Code of Practice of UK NARICs outlines that, in order to recognise prior learning and degrees, they use ”the evaluation of learning outcomes achieved through all paths and progression routes” [112]. This enables the recognition of national, international and joint qualifications at a deeper level, and considers not only the formal features like credits, but also the knowledge and ability that a student gains on completion of his or her studies. For this reason, priority is given to outcomes-based approaches of evaluation. The Code of Practice notes that it also considers such criteria as the status of the awarding institution and of the qualification within the country’s education system, in addition to the course content, structure and duration of the educational course and the methods of study and examination. The UK NARIC Band Framework is used for the resulting evaluation. The Framework consists of sixteen bands and the generic outcomes assigned to them. This hierarchy enables the alignment of international degrees with the British hierarchy of higher education qualifications.

The Russian NARIC redirects the applicants seeking recognition of their qualifications to an organisation called ”Glavexpertcentre”. Its responsibility is the nostrification of international degrees. Unfortunately, little information is provided regarding the internal processes of the organisation inside or the criteria that are used. The applicant submits his or her original qualifications with a

translation into Russian, the transcripts thereof and a copy of his or her passport. S/he then awaits a decision from the authorities.

Both the British and the Russian NARICs use the information management system (IMS) for the support of their activities. However, neither of them is reported to use automation, and particularly not for the process of degree or prior learning recognition. The UK's IMS is described as a large database, containing up-to-date information regarding international education systems. The Russian IMS enables tracking the status of the application online.

2.2.5 Documenting Courses in the UK and the RF

The research showed that higher education qualification recognition is based on a comparison of the specialised supporting documentation that contains a description of the educational courses and the relevant modules. Competencies and learning outcomes are of primary importance. A study that documented educational courses and modules in universities in the United Kingdom and the Russian Federation was conducted. Special attention was paid to the representation of the learning outcomes and competencies.

Documenting Educational Courses in the UK Universities

The United Kingdom has independent organisations that are responsible for ensuring the quality of higher education and its ability to satisfy the demands of industry. In order to fulfil their tasks, these bodies produce packages of information that include the requirements for documentary support of the educational processes at higher education institutions.

A programme specification (PS) is a document that contains information regarding an educational course taught at a university. Despite different titles being used to refer to this type of document, such as course programme, course specification and course template, in this work we will use the term employed by the Quality Assurance Agency for Higher Education in the United Kingdom (QAA).

The reason for using the terminology given by the QAA as a standard is that it is the most powerful and widely recognised independent organisation, the main

responsibility of which is to ensure the quality of and to encourage improvement of the standard of education provided by British higher education institutions. This aim is achieved by regular reviews and audits of the universities and other higher education bodies in order to examine the quality of the learning opportunities they offer to students and the academic standards of the awards they make.

Guidelines for Preparing Programme Specifications

The Guidelines for Preparing Programme Specifications issued by the Quality Assurance Agency declare that a "programme specification is a concise description of the intended learning outcomes from a higher education programme, and how these outcomes can be achieved and demonstrated" [97, p. 9]. The definition is expanded by stating that the programme specifications are " ... the definitive publicly available information on the aims, intended learning outcomes and expected learner achievements of programmes of study" [98, p. 8]. It is important to note that the QAA does not prescribe any specific format for presenting the programme specifications, but merely suggests the information that should be included. However, it is recommended that a university should have internal standards for writing the PSs, which will help to keep the documents consistent.

The Guidelines for Preparing Programme Specifications also indicate the necessity of following the recommendations given in the Framework for Higher Education Qualifications in England, Wales and Northern Ireland (FHEQ, issued by the QAA) and the National Occupational Standards.

The FHEQ was prepared by the QAA "to inform international comparability of academic standards [...] and to facilitate student and graduate mobility". In this way, it supports the main goals of the Bologna Declaration and thus facilitates the integration of the UK universities and other HE providers into the European Higher Education Area. FHEQ aligns the typical British higher education qualifications and levels with those of the Bologna educational cycles. It is important to note that the FHEQ contains descriptors for higher education qualifications of all levels, in the form of generic competences.

The programme specifications from the De Montfort University (Leicester, UK) were studied as an example of the implementation of the QAA's recommendations.

A DMU PS contains the sections Basic Information, Entry Requirements and Profile, Course Description, Outcomes, and Structure and Regulations.

The section Basic Information consists of the course title, the internal code, the level (undergraduate or postgraduate), the faculty, the department, the geographic location, a list of all possible exit awards - including the highest, the mode of attendance and the course leader. The Entry Requirements and Profile outline the conditions that an applicant is required to meet. The Course Description includes the characteristics and aims of the course and describes the teaching, learning and assessment strategies. The Outcomes section includes generic course outcomes that are divided into several groups. The section Structure and Regulations outlines the educational modules included in the educational course. Each of the modules is characterised by an assigned number of credits, the students' compliance for registration, the semester(s) during which it is taught and the geographic location of studies.

Documenting Educational Modules in UK Universities

British universities have a certain level of flexibility in documenting the educational modules, as the Quality Assurance Agency does not restrict this process, and nor do the Bologna recommendations. Thus, in this research, we analysed the module templates produced by the De Montfort University (Leicester, UK) as an example.

The structure of a module template is unified throughout the university. It contains three main sections, namely the Basic Module Information, the Module Definition and the Module Delivery Variations.

The basic data provide the module's title, internal code, ECTS credit value, DMU credit value, faculty, module leader and the module's pre-requisites, if any.

The "Module Definition" consists of the following sections.

1. Module Characteristics.

   This subsection provides a general description of the educational module.

2. Learning Outcomes.

   This subsection contains the list of the module's learning outcomes.

3. Learning and Teaching Strategies.

   This subsection introduces the types of activities that will be performed during the studies, such as lectures, laboratory work and tutorials.

4. Key Skills.

   This subsection describes the skills that the students will gain on completion of the module at a general level. For example, it could state that a student will be able to compile presentations or to write reports.

5. Module Syllabus.

   This subsection provides an overview of the topics covered by the module.

6. Module Keywords.

   This subsection contains the module's main keywords.

7. Assessment and Reassessment Rationale.

   This subsection clarifies the means that will be used to assess the students' performances. For example, they may include coursework and examinations. Reassessment strategies are also mentioned; for instance, the student may be re-assessed in only the components that s/he failed.

8. Resources.

   This subsection covers funding information, the number of student places available and the cost of study. It may also describe the learning resources used, such as the library and additional services.

9. Quality Assurance.

   This subsection describes the policy used to collect students' opinions of the module. The students may be asked to provide feedback via module questionnaires.

The section Module Delivery Variations is only completed if the module is offered in different ways to distinct groups of students. For example, it may be adapted for use in several educational courses. The Module Delivery Variations section describes ways in which the information provided in the sections Basic Modules Information and Module Definition is adapted to suit several scenarios.

Documenting Educational Courses in the RF Universities

Documenting of educational courses and modules in the Russian Federation provides less flexibility for the universities when compared to the UK's higher education institutions. The structure and content of the supporting documentation is restricted by the Ministry of Education and Science and several other approved authorities.

Federal State Educational Standards for Higher Professional Education

The Federal State Educational Standards for Higher Professional Education (FSES HPE) regulate contents, characteristics, learners' competencies on completion of their studies and many other aspects of an educational course. There are separate FSESes for the educational programmes of each level and each professional area of study, for example a Bachelor's Degree in Computer Science and Engineering. If a university fails to comply with the requirements of the corresponding FSES, the Ministry of Higher Education and Science will not provide a licence for teaching the course.

FSESes for HPE (e.g. [87, 88]) contain a generic description of what is expected from the educational modules included in an educational course. This section is called the Structure of the Main Educational Programme and is organised in the form of a table. The obligatory, expected learning outcomes of the modules are classified in accordance with the types of activities, for example the humanities, science and technology, specialisations and so on, in the first column. The other columns present the sample module titles, the number of credits and the references to generic competencies that are realised by the specific learning outcomes.

Standard Main Educational Programmes

In obedience to the corresponding FSES for HPE, a higher education institution designs a Sample Main Educational Programme (SMEP). This document gives the main characteristics of an educational course. It lists all the professional fields in which the graduate will be able to work, such as science and research, management and social occupations. The SMEP may also contain a description of the profiles within the professional area of studies - for example, the area of applied mathematics and computer science may contain such profiles as mathematical modelling, systems analysis, cybernetics and various others. A set of topics and

keywords is assigned to each profile. A separate, large section contains all the competencies that a learner will be able to perform on completion of his or her studies. Profile-dependent competencies are grouped separately.

The SMEP (e.g. [17, 89]) includes a section called the Approximate Study Plan (ASP). The ASP is a table that contains the entire list of educational modules (disciplines) for the course. The ASP is a specification of and an expansion to the Structure of the Main Educational Programme in the FSES of HPE. Similar to the expected learning outcomes of the modules in the Structure of the Main Educational Programme, the educational modules in the ASP are classified according to the types of activities. The modules of each type of activity are divided into Basic Components and Variative Components. The basic educational modules realise the obligatory, expected learning outcomes of the Structure of the Main Educational Programme. A student must take all the disciplines in the Basic Components. The variative section contains the modules that a particular higher education provider wants to include in the programme. In turn, these disciplines are divided into two subcategories: compulsory, which a student must take, and optional, from which a learner may select one or more, if any. An approximate study plan contains columns such as the module title, workload in credits and notional hours, approximate distribution across semesters, types of assessment and references to generic competences that are realised by the learning outcomes of the module.

The largest section of the SMEP contains annotations to the modules. Each presents the title, the aims and objectives of the disciplines, and lists the main topics and keywords.

Documenting Modules in the RF Universities

Each educational module taught at a university should be described in a document called the Approximate Programme of a Discipline (APD). The structure of the APDs may differ slightly from one university to another, particularly at present during the reformation of the higher education system and revisions to its documentary support.

Currently, an approximate programme of a discipline contains the following sections.

1. Aims and objectives.
2. The lists of modules that a student should have already completed (pre-requisites) and those which s/he will be able to take on completion of this discipline.
3. The learning outcomes of the disciplines. At present, these are divided into knowledge, abilities and skills. The LOs refer to generic competencies of the educational courses.
4. Contents of the module.

   This section contains the list of topics and associated keywords that will be taught during lectures. If the discipline incorporates seminars, practical work (such as course projects or laboratory work) or self-study, these are also described in detail. Each of the activities refers to the number of topics in the lectures. The APD restricts the number of notional hours needed to perform each of the tasks. Many modules include several assessment measures that will be employed during the process of learning. These may include written homework and tests. The APD may contain examples of the tasks and schedule them according to the weeks in the semester.
5. Methodical Materials.

   This section provides a list of advised information sources for studies, such as books, articles and the like.
6. Methods and organisation of the educational process.

   This section contains a description of the teaching and learning strategies, as well as the requirements for the material support of the education process, such as equipment and software for laboratory work.

2.2.6 Comparison of Competence Models in British and Russian Educational Systems

Referring to the Bologna recommendations, the countries participating in the process should "elaborate a framework of comparable and compatible qualifications for their higher education systems, which should seek to describe qualifications in terms of workload, level, learning outcomes, competences and profile" [22, p. 13].

This advice, and the approach accepted by the ENIC-NARIC Network, vividly outlines the significant role of competencies and learning outcomes for the recognition of higher education qualifications which, in practice, includes a comparison of educational courses. This, together with the current tendency towards student-centred education, provides the grounds on which to base a comparison of the educational course of the universities when matching their competencies and learning outcomes. In this work, we focused on the competence models in higher education systems in the United Kingdom and in the Russian Federation as an example.

The Model of Competencies in the British Higher Education System

The hierarchy of competencies in the British higher education system is presented in Figure 2.1.

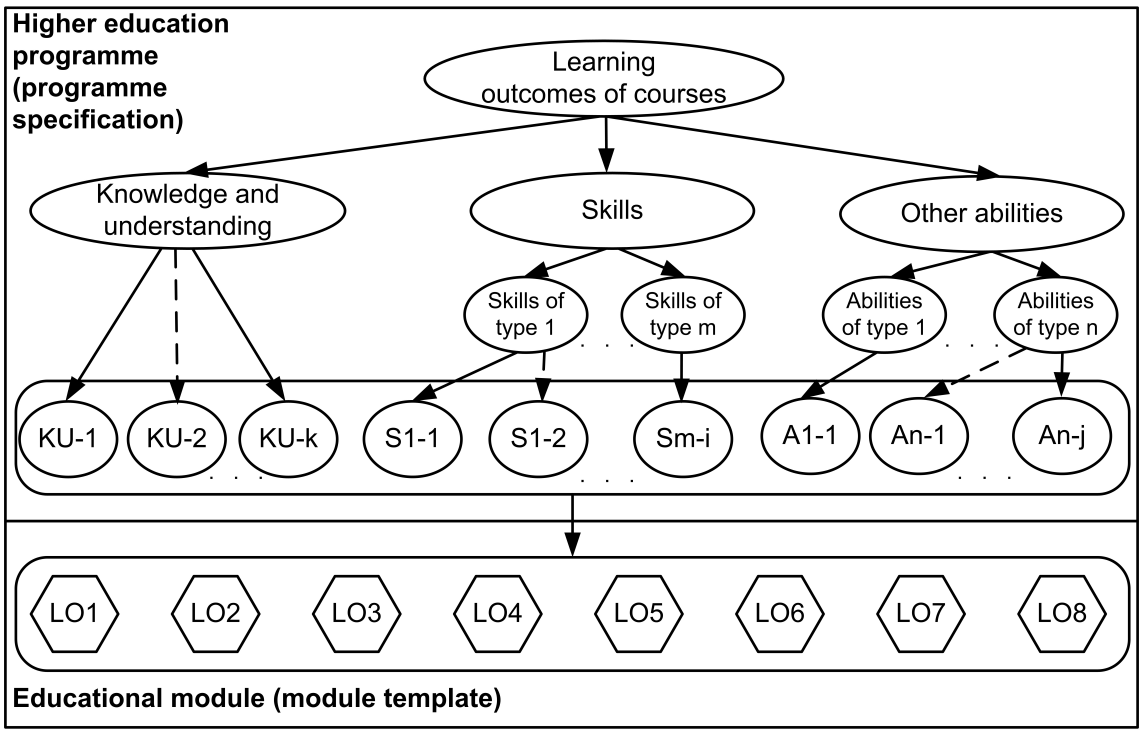

Figure 2.1: The Hierarchy of Competencies in the British Higher Education System

The part of the scheme in the upper rectangle in Figure 2.1 illustrates the outcomes of educational programmes at course level. The classification is given in the form in which it may be present in a programme specification. All the generic course's

outcomes can be divided into several sets, namely knowledge and understanding, skills, and other abilities. In practice, the last group, other abilities, may be substituted for by several other types of competencies. The skills and other abilities, or other types of competencies, can be further split into other subtypes (S1-1 to Sm-i and A1-1 to An-j in Figure 2.1). Each group of the course's outcomes includes several generic competences (KU-1 to KU-k, S1-1 to Sm-i and A1-1 to An-j in Figure 2.1). These can either be compulsory (linked with solid line arrows in Figure 2.1) or optional (linked with dashed line arrows in Figure 2.1). This depends on whether they are realised in obligatory or optional educational modules.

The lower rectangle in Figure 2.1 includes the intended learning outcomes of the modules (hexagons marked from LO-1 to LO-8). The British programme specifications and module templates do not contain references that link course outcomes to the intended learning outcomes of the modules. Thus, we cannot judge how a generic course's competencies are realised in particular educational modules. In the picture, 2.1 the entire set of course outcomes is linked by a solid arrow to the complete set of the modules' learning outcomes. Both sets are incorporated into rounded rectangles in the upper and lower parts of the picture 2.1, respectively.

The Model of Competencies in the Russian Higher Education System

The hierarchy of competencies in the Russian higher education system is presented in Figure 2.2. The taxonomy complies with the hierarchy in the Federal State Educational Standards for Higher Professional Education, the Sample of Main Educational Programmes and the Approximate Programme of Disciplines.

The section of the scheme in the upper rectangle in Figure 2.2 illustrates the outcomes of educational programmes at the level of higher education courses. In the FSES of HPE of the Russian Federation, these are called competencies. Normally, each SMEP contains at least two classes of competences: general cultural and professional. The first type describes the generic knowledge and abilities of a learner at a somewhat abstract level, such as being able to self-develop in intellectual, physical and cultural areas and then apply these results to professional work. The second type includes competencies in the student's professional field. However, it operates by using generic terms of the data domain of the degree. For

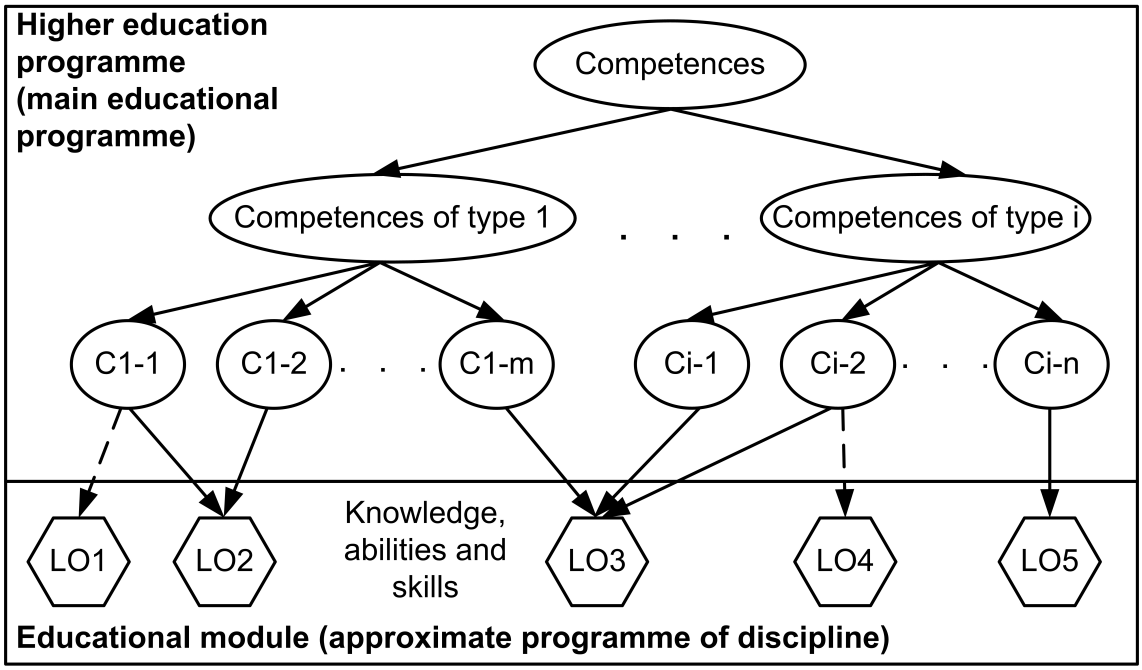

Figure 2.2: The Hierarchy of Competencies in the Russian Higher Education System

example, a statement would more likely be phrased as like ”be able to develop automated information systems” than as ”be able to design a database in RDBMS Oracle and program in Java”. Apart from the main types, the classification may also include competencies in scientific research, manufacturing and technology, pedagogy, organisational and managerial competencies and other kinds of activities.

The lower rectangle in Figure 2.2 shows the learning outcomes of educational modules. In the Approximate Programme of Disciplines, they are divided into knowledge, abilities and skills. Each LO in the APD has a reference to the course competence code(s) that it realises. Therefore, we will always know the particular contents of the sets of LOs associated with each course outcome. Each competence can be referenced in several LOs of one or many APDs. As an educational module may be either obligatory or optional for the student, a learning outcome follow suite. In Figure 2.2, the solid line arrows link the competencies to the LOs of the compulsory modules, while the dashed line arrows connect the LOs of optional modules.

2.3 Ontologies, Similarity Measures and Ontology Alignment

2.3.1 Knowledge Representation Models

There is currently a wide variety of knowledge representation models. They differ in terms of expressiveness, formalism, and the ability to represent fuzziness and to perform inference.

In [16, p. 147-156], Bashmakov outlines the following classes of knowledge representation models.

1. **Logical models**

   A logical model is a formal system (FS), presented as the following tuple.

   $$FS = (T, P, A, R) \tag{2.1}$$

   In the tuple 2.1, **T** stands for the set of terminal symbols used to build the expressions. **P** is the set of syntactical rules for building structurally correct expressions from the terminals. **A** is the set of axioms, which are the a priori correct expressions of the formal system. The set **R** contains the inference rules, which can be used to convert some syntactically correct expressions to others. The logical models include the propositional and predicate calculi.

   (a) **Propositional calculus**

   Propositional calculus is the simplest logical model. The axioms from the set A (Formula 2.1) form the basis thereof. The system of axioms must be consistent, complete and independent. The set of rules, R, contains two types of rules. The first is, if X and $X \rightarrow Y$ are true, then Y is true. The second is the possibility to substitute one statement with another in a syntactically correct formula. However, poor expressiveness is a disadvantage of propositional logic.

(b) **Predicate calculus (First order logic)**

The predicate calculus evolved as a result of the evolution of propositional calculus. A predicate is a function that reflects the properties of or relationships between objects. The arguments of these functions are called terms. They correspond to the object of the properties and relationships. The value of the function can be either true or false. The alphabet of the first order logic (FOL) extends the alphabet of the propositional calculus with predicates, variables, constants, and universal and existence quantifiers. The sets of axioms and rules are based on the corresponding sets of propositional calculus.

First order logic is the basis of the widely used relational databases. The family of description logics, which is a subset of first order logic, provides the formalism for ontology representation languages.

The high level of formalism, which provides possibilities for logical inference, consistency, and the use of the same means for knowledge representation and the processing thereof are the main advantages of the logical models. However, the poor readability and restrictions to expressiveness for the sake of consistency are among their drawbacks.

2. **Production models**

A production model can be presented as the following tuple [16, p.150].

$$PM = (W_i, U_i, P_i, A_i \rightarrow B_i, C_i) \tag{2.2}$$

Here, W_i is the field of application of the i-th production, U_i is the precondition for the i-th production, including such information as its priority compared to other productions. P_i is the condition that allows the firing of the production, $A_i \rightarrow B_i$ is the core of production, which is read as "if . . . then . . . " and C_i is the post condition, which dictates the changes made to the system after the firing of the production.

A system that is based on the production model contains the base of declarative facts regarding the data domain, the base of productions (rules) and the interpreter, which analyses the preconditions and controls the sequence of the productions' application.

The advantages of a production model are the clear and visual interpretation of separate rules and the ability to easily enrich the knowledge base. However, the complexity of inference, ambiguity when choosing between rules with the same priority and the complexity of dealing with the interconnections between the productions are among the weaknesses of production-based systems.

3. **Frame models**

   According to Bashmakov [16, p.151], a frame is a data structure that describes an object or a situation in a domain of interest. Each frame contains an identifier and a slot, which contains its attributes and may link it to another frame in the system. The frames can be stored in a taxonomy with is-a relations, which enables inheritance. Inference in a frame model is based on the message exchange between the frames.

   Visibility, homogeneity and the high level of data structuring are the strengths of the frame modelling. However, inference in a frame model is highly complicated and lacks efficiency.

4. **Network models**

   Network models represent a domain of interest as a set of objects (nodes) and the relationships between them (arches). The nodes and the arches are marked with the terms from the data domain. The types and rules of inference that can be performed across the network depend on the types of relationships that link the objects. For example, a semantic network can represent a simple taxonomy if it contains only is-a relationships, or a scenario if it includes cause-effect relationships. One semantic network can combine several types of relationships.

   The high degree of commonality and visibility, the ability to represent many types of relationships among the data domain objects and the high degree of expressiveness are the advantages of network models. However, all these positive aspects lead to complexity in the creation of a unified reasoner across a model.

Ontology belongs to the class of combined models, because it integrates the semantic network and either the frame model, the logical model, or both. The first ontologies and ontology languages were based on frame models. Modern

ontology representation languages are based on the family of description logics, which are subsets of first order logic. The main difference between frame- and DL-based ontologies is the following. The frame model can only store explicit statements about data domain. A DL-model may also contain logically inferred knowledge.

Thus, a modern, DL-based ontology combines the features of two powerful data models. On one hand, ontology is a semantic network in which the classes and instances are nodes, while the relationships between them are the edges. On the other hand, it is based on a logical model. Thus, ontology inherits the advantages of both approaches by being formal and supporting inference on one hand, yet offering a high level of semantic expressiveness and visible representation on the other.

Currently, ontologies are utilised for numerous tasks, such as the automatic composition of services [115], question answering systems [75, 99], descriptions of documents and data domains [13, 82, 96, 101, 107], overcoming semantic heterogeneity in information systems [69, 118] and semantic searches [27, 68]. The direction called Ontology Driven Software Engineering (ODSE) recommends using ontologies at different stages of the software engineering process [116].

When applied to the field of education, ontologies are used to develop e-learning systems [11, 102], to prescribe courses for potential students [65] and to describe the data domains of higher education and learning [47, 120].

2.3.2 Informal Definition of Ontology

Computer science has borrowed the term "ontology" from philosophy. When applied to philosophy, ontology means the study of existence and a system of worldviews.

In information technologies, the first and most popular variant of the definition of ontology is the one proposed by Gruber (1993), who stated that ontology is an "explicit specification of conceptualisation" [44].

Here, the term "conceptualisation" reflects the process of the translation of the natural language representation of a data domain to its specification in a formal,

machine-readable language. Conceptualisation is also understood as a description of a number of concepts of a data domain, knowledge regarding them and the relationships between them. In other words, ontology is a formal system that consists of a set of concepts and a set of assertions about these concepts, based on which classes, objects, relations, functions and theories can be built. "Explicit" means that the concepts and restrictions on them should be defined explicitly and unambiguously.

The definition of ontology was later expanded as follows: an ontology is the "explicit, formal specification of a shared conceptualisation of a domain of interest" [39, p. 12]. "Formal" means that an ontology should be represented in a machine-readable language. "Shared" reflects the idea that an ontology captures consensual knowledge that is not limited to an individual. "Shared" does not necessarily imply being shared globally, but the fact that the meanings of the concepts that are provided in the ontology are accepted by a group of people. The reference to a domain of interest indicates that, with regard to domain ontologies, the aim is not to model the entire world, but rather to model only those parts of a certain domain that are relevant to the task.

According to [79], ontology is a specification of conceptualisation, but only in the aspect that depends on a particular field of interest. Irrespective of the type of ontology model, it should include a vocabulary of terms and definitions thereof. This approach restricts the possible interpretations of the concepts' definitions and enables the consideration of relationships between them. In this case, the definition of ontology intersects with those of well-known thesauri.

Takeda et al. [108] placed ontology at the centre of the knowledge organisation problem, as each data domain may include different definitions for the same terminology. In this case, ontology is used to structure the information as a mediator between human and machine-oriented knowledge representation. Here, ontology is defined as agreement on a mutual understanding of a domain of interest for achieving a particular goal.

According to [45], ontology should characterise conceptualisation and should restrict the possible values of predicates and functions in order to agree on knowledge representation in a certain logic-based language. In this view, ontology refers to a logical theory in which axioms restrict the interpretation of non-logical symbols of the language.

Thus, informally, ontology is a description of the system of views concerning a data domain as applied to a certain task. An ontological description includes the terminology and the imposed rules that restrict the definitions and the relationships between the terms. Formally, ontology is a system of concepts and a set of assertions, based on which a system of classes, objects, relations and inferences can be built.

2.3.3 Formal Definition of Ontology

In [42, p.39] Euzenat and Shvaiko defined ontology as the following tuple:

$$O =< C, I, R, T, V, \leq, \perp, \in, =>, \tag{2.3}$$

where

C is the set of classes used to store the sets of individuals in a domain of interest,

I is the set of individuals, which are particular objects in the data domain of interest,

R is the set of binary relations, either between two individuals (known as Object property), or between an individual and a data type (known as Data type property),

T is the set of data types (for example, integers, strings)

V is the set of particular values (C, I, R, T, V being pairwise disjoint),

$\leq$ is a relation on $(C \times C) \cup (R \times R) \cup (T \times T)$, called specialisation,

$\perp$ is a relation on $(C \times C) \cup (R \times R) \cup (T \times T)$, called exclusion,

$\in$ is a relation over $(I \times C) \cup (V \times T)$, called instantiation,

= is a relation over $I \times R \times (I \cup V)$, called assignment.

This definition includes the concepts of classes, objects and data type relations. It enables the representation of the classes' taxonomies and hierarchies of properties, the instantiation of classes and the assignment of relations. At the same time, it is easily readable and clear in terms of human understanding. Furthermore, this definition lacks the axiom component that appears in formal, logic-based ontologies. Axioms consist of logical statements that are always true, and the knowledge that can be derived from them. They may contain ontology restrictions (constraints) that are imposed on the values of properties. The types of constraints depend on the expressiveness of the ontology representation language. For example, a constraint may state that an individual of class c_1 must be related through relation r_1 to one and only one individual of class c_2. Axioms also include the formal rules (if-then statements) that describe logical inferences across ontology. For example, they can state that if an individual is of the class c_1, it must also be an individual in class c_2.

We utilised the formal definition of ontology proposed by Euzenat and Shvaiko (2007) and extended it via the concept of axioms. Thus, our ontology can be presented as the following tuple:

$$O =< C, I, R, T, V, A, \leq, \perp, \in, => , \tag{2.4}$$

In the definition 2.4, the symbol "A" stands for axioms, while all the other constituents have the same meaning as they do in the model proposed by Euzenat and Shvaiko (2007).

2.3.4 Description Logics

Description logics are a family of mathematical logics and are a decidable subset of first-order logic [111]. It is a set of knowledge representation formalism that represents a data domain by defining the concepts and terminology of the domain of interest, its objects and individuals. Description logics provide formal, mathematically based, semantic definitions of concepts. It is suited to reasoning, which allows for making logically proved assertions using statements that already exist. A knowledge representation system based on description logic enables the creation of knowledge bases, reasoning regarding them and the management thereof. Description logics are widely used for ontology modelling [73].

The basic attributive description logic language is called the Attribute Language Concept (ALC). In the ALC, concepts are defined with the help of operators union $\cap$, intersection $\cup$, negation $\neg$, universal $\forall$ and existential $\exists$ quantifiers [73, p. 12].

Other description logics' names are constructed as sequences of the following symbols:

S - ALC, enriched with transitive roles;

H - role hierarchy;

R - according to [73, p. 13], "most commonly refers to the presence of role inclusions, local reflexivity Self, and the universal role U, as well as the additional role characteristics of transitivity, symmetry, asymmetry, role disjointness, reflexivity, and irreflexivity";

O - nominals;

I - inverse roles;

N - not-defined number constraints;

Q - qualified number constraints;

F - functional constraints;

D - data types. The letter "D", which stands for 'data types', is normally placed in brackets at the end of the abbreviations of the names of DLs.

For example, description logic SHOIN (D) extends ALC with transitive roles, roles' hierarchy, nominals, not-defined number constraints and data types.

2.3.5 Ontology Languages and DL

OWL

The OWL (Web Ontology Language) is a language for defining and instantiating Web ontologies [8]. According to Horrocks [56, p.8], it was strongly influenced by description logic, the frames paradigm, the Resource Description Framework

(RDF) and OWL's predecessor, the DARPA Agent Markup Language, and the Ontology Inference Layer (DAML+OIL) language. The specification for OWL was created in 2004.

RDF is a data model for storing metadata of a specified structure as an additional page or block inside each web page. In RDF, facts are represented as triples (Subject, Predicate, Object), where subjects, predicates and objects are the names of real world entities [74, 109]. DAML+OIL is a frame-model and DL-based ontology-representation language, which was created from the DARPA Agent Markup Language (DAML) and the Ontology Inference Layer (OIL). The DAML+OIL operators can be unambiguously turned into expressions in the description logic SHIQ [15, 52].

The OWL language provides three increasingly expressive sublanguages designed for use by specific communities of implementers and users [105].

OWL Lite supports those users who primarily need a classification hierarchy and simple constraint features. It should be simpler to provide tool support for OWL Lite than for its more expressive relatives, and to provide a quick migration path for thesauri and other taxonomies.

OWL DL supports those users who want the maximum expressiveness without losing the computational completeness (all entailments are guaranteed to be computed) and the decidability (all computations will finish in finite time) of reasoning systems. The OWL DL includes all OWL language constructs with restrictions such as type separation (a class cannot also be an individual or a property, and a property also cannot be an individual or class). OWL DL is so named because of its equivalence to description logic SHOIN (D) [53].

OWL Full is suitable for developers who want to gain maximum expressiveness and the syntactic freedom of RDF with no computational guarantees.

Each of these sublanguages is an extension of its simpler predecessor, both in terms of what can be legally expressed and of what can be validly concluded. The following set of relations hold. Their inverses do not.

1. Every legal OWL Lite ontology is a legal OWL DL ontology.
2. Every legal OWL DL ontology is a legal OWL Full ontology.
3. Every valid OWL Lite conclusion is a valid OWL DL conclusion.
4. Every valid OWL DL conclusion is a valid OWL Full conclusion.

OWL 2

In 2006, Horrocks, Kutz and Sattler introduced description logic SROIQ and proposed to utilise it to improve OWL [54]. They claimed that the first version OWL lacked such important expressive means as disjoint, reflexive, irreflexive, and antisymetric roles, and negated roles assertions, role inclusion axioms and a universal role. The DL SROIQ offers all these possibilities while remaining decidable [54].

The specification for OWL 2 was developed in 2009.

Historically, OWL was based both on description logics and on RDF; thus, two different, yet related, ways of assigning meaning to OWL 2-ontologies exist. They are described in the documents "OWL 2 Web Ontology Language RDF-Based Semantics" and "OWL 2 Web Ontology Language Direct Semantics", respectively.

The Direct Semantics of OWL 2 are compatible with Description Logic SROIQ and is thus called OWL 2 DL [55]. It subsumes the language OWL-DL. The RDF-based semantics of OWL 2 are called OWL 2 Full. OWL 2 Full is undecidable. Thus, there are several reasoners that cover OWL 2 DL fully, but none for OWL 2 Full [48].

OWL 2 DL contains three profiles, OWL 2 EL, OWL 2 QL and OWL 2 RL [25]. Each of the profiles is a simplified version of OWL 2 DL and is suited for a particular purpose. OWL 2 EL is suited for ontologies containing a large number of classes and properties. OWL 2 QL is aimed at applications containing many instances and is aimed at answering queries. OWL 2 RL is suited to applications that require scalable reasoning without reducing expressiveness. Reasoning can be performed using Rule Language.

In our work, we utilised the OWL 2 DL language to represent ontologies. Neither OWL-DL nor any of the OWL 2 DL profiles are applicable, as they do not support qualified number restrictions. At the same time, OWL 2 DL is sufficiently expressive not to sacrifice decidability to using OWL Full or OWL 2 Full.

2.3.6 Semantic Web Rule Language

The Semantic Web Rule Language (SWRL) is based on a combination of OWL DL and Rule Markup Language. It enables the enrichment of a DL ontology with the rules produced in accordance with Horn logic.

A typical rule in Horn logic has the following form.

$$a_1 \wedge a_2 \wedge \cdots \wedge a_n \rightarrow b \tag{2.5}$$

The left part of the rule is called the antecedent, or body, and contains the conjunction of atoms. The right part of the rule includes the consequent, or head. If all the atoms in the body of the rule hold, then the consequent is also true. The head of the rule cannot include any variables that are not present in the body.

According to [51], an atom can be a unary or a binary predicate [46, p. 234] and is defined in Backus-Naur Form (EBNF) as follows:

$atom ::= ① \ description\ (i-object)$
$| ② \ dataRange\ (d-object)$
$| ③ \ individualvaluedPropertyID\ (i-object,\ i-object)$
$| ④ \ datavaluedPropertyID\ (i-object,\ d-object)$
$| ⑤ \ sameAs\ (i-object,\ i-object)$
$| ⑥ \ differentFrom\ (i-object,\ i-object)$
$| ⑦ \ builtIn\ (builtinID\ \{d-object\})$

Here builtin ID means a URI reference to a built-in SWRL function. I-object stands for an individual variable or a URI of an individual in an ontology, while 'd-object' identifies either a data variable or a data literal. SWRL has three syntaxes: abstract, and based on XML and RDF. For the sake of brevity, we will give the examples in the abstract syntax. In abstract syntax, variables are preceded by a question mark (?), an arrow ($\rightarrow$) means implication and a conjunction symbol ($\wedge$) is used to separate atoms.

The definition above means that an atom may hold in one of the following cases.

1. description stands for the URI of a class in an ontology. For example, Module (Databases) means that individual Databases belong to the class Module.

2. dataRange means a data range. For example, int(?y) means that ?y is of data type &xsd:int.

3. individualvaluedPropertyID means, that two i-objects are related by an object property. For example, hasModule (?x, ?y) means that individual variables ?x and ?y are connected through the relation hasModule.

4. datavaluedPropertyID means that an individual i-object has a data type property with the value d-object. For example, ModuleCredit (Databases, 5) means that individual Databases have the property ModuleCredit with the value of 5.

5. sameAs holds if two i-objects are interpreted as being equal. For example, sameAs (?x, ?y) means that individual variables ?x and ?y stands for individuals, who are interpreted as being equal.

6. differentFrom holds if two i-objects are interpreted as being different. For example, differentFrom (Databases, Salad) means that individuals in Databases and Salad are interpreted as being different.

7. builtIn holds if the SWRL built-in relation is true for a d-object. SWRL built-in is a function over one or more operands, which returns either true or false values. Built-ins are divided into several classes in accordance with their usage: math, string, date/time and others. They are helpful in cases when the ontologist needs to know if a condition, which cannot be expressed in the terms of ontology, holds. For example, builtIn (swrlb:subtract (?x,?y,?z)) holds if and only if ? x = ? y - ? z.

Despite the significant advantages, the use of the rules has a serious drawback, which is the introduction of undecidability into an ontology. For example, the following rule is undecidable.

$$Course(?x) \wedge int(?y) \wedge numModules(?x, ?y) \wedge int(?z) \wedge swrlb : add(?z, ?y, 1) \rightarrow numModules(?x, ?z) \tag{2.6}$$

The intended aim of the rule 2.6 is to increase the number of modules that an educational course contains by one (data type property numModules). However,

firing this rule will cause an endless recursion, due to the repeatability of the rule's execution. To avoid such cases, the concept of DL-safe rules was introduced. A DL-safe rule binds the individual variables only to the explicitly stated individuals of an ontology. Another safety condition requires that each data variable in a data range atom of the antecedent also occurs in a data property atom in the body of the rule [43]. The rule 2.6 does not satisfy the second safety condition, because the data variable ?z does not occur in any data property atom.

SWRL-rules are used for the following purposes [51], [46, p. 233].

1. Inheritance of data type and object property values. For example, if an educational course contains a module that has a learning outcome, then the course contains the learning outcome as well.

   Example 1. Object Property

   $Course(?x) \wedge Module(?y) \wedge LearningOutcome(?z) \wedge$

   $hasModule(?x, ?y) \wedge hasModuleLearningOutcome(?y, ?z)$

   $\rightarrow hasCourseLearningOutcome(?x, ?z)$

   This example illustrates the inheritance of object property value. The rule says that if an educational course has a module that has learning outcomes, then the course will also have them.

   Example 2. Data type Property

   $Course(?x) \wedge Module(?y) \wedge string(?z) \wedge ModuleDataDomain(?y, ?z) \wedge$

   $hasModule(?x, ?y) \rightarrow CourseDataDomain(?x, ?z)$

   This example illustrates the inheritance of object property value. The rule says that an educational course has all the data domains to which its modules are assigned.

2. Assertion that a combination of certain statements implies another one. For example, if an educational course contains an educational module that a student may take only if s/he wishes, then this module is optional.

   $Course(?x) \wedge Module(?y) \wedge hasModule(?x, ?y) \wedge Student(?z) \wedge$

   $takesCourse(?z, ?x) \wedge mayTake(?z, ?y) \wedge \rightarrow Type(?y, optional)$

3. Axiomatising of unique name assumptions. For example, if one educational module is called Databases and another is called Data models, these are two different disciplines.

   $Module(?x) \wedge Module(?y) \wedge Name(?x, Databases) \wedge Name(?y, DataModels)$

   $\rightarrow differentFrom(?x, ?y)$

2.3.7 Knowledge Representation System based on DL

The architecture of a knowledge representation system based on description logic is presented in the following figure, adopted from [14]. We improved the original image by adding an RBox that contains rules to the knowledge base. This was done to represent a Knowledge Representation System based on DL SROIQ, as in previous versions of the scheme the Knowledge base consisted of TBox and ABox only [54].

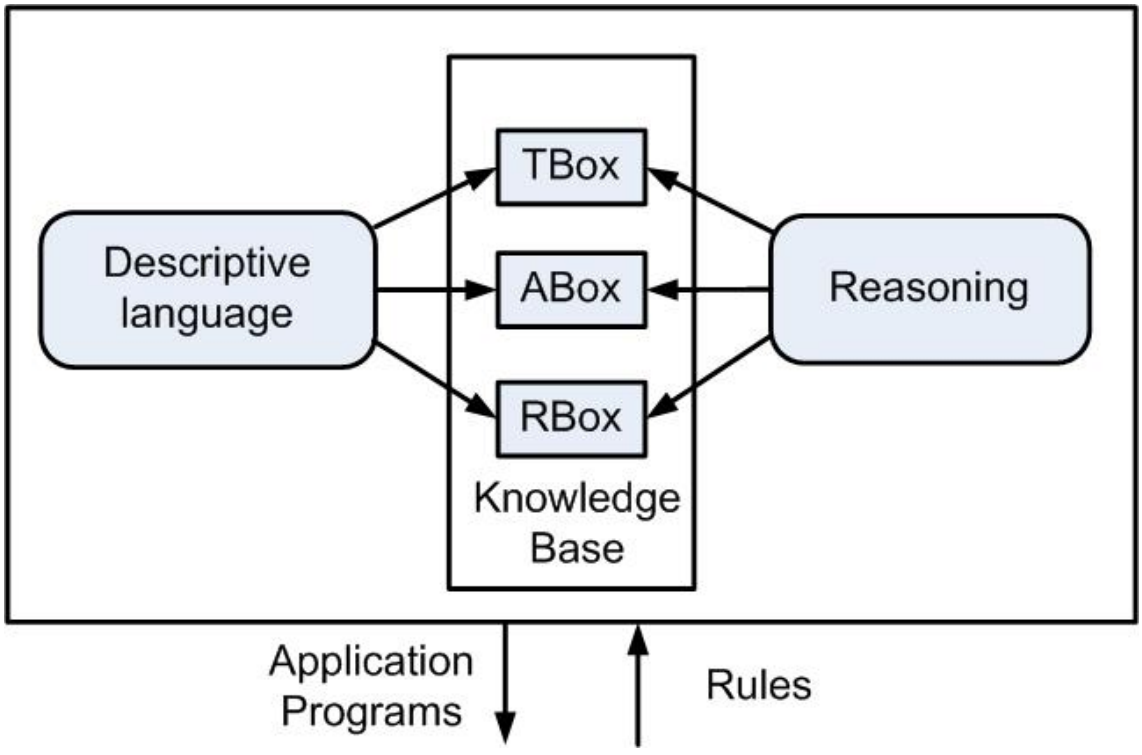

Figure 2.3: Architecture of a Knowledge Representation System based on Description Logic SROIQ

1. Knowledge base. Ontology O is mapped into DL SROIQ-knowledge base $KB =< TBox, ABox, RBox >$, where:
   (a) TBox is a knowledge base component that introduces the terminology (the vocabulary of an application domain). TBox is a set of axioms that includes concept inclusion and equivalence.
   (b) ABox is a knowledge base component that contains assertions about named individuals in terms of vocabulary. ABox is a set of axioms that includes concept and role instantiation.
   (c) RBox contains a hierarchy of roles (or role inclusions) and role assertions (transitivity, reflexivity etc.).
2. Description language is one of the formal languages that are based on one of description logic.
3. Reasoning is a procedure of making logically provable assertions about the terminology.
4. Application programmes are information systems that work with the knowledge base.
5. Rules are an extension of the logical core formalism, and can still be interpreted logically.

Assertions in TBox, RBox and ABox can be represented as the first-order logic formulae. A description logic-based system can not only store them, but can also reason using them.

2.3.8 Ontology Reasoning

The OWL language was deliberately based on description logic to enable ontology reasoning [56, p.22]. This ability allows for making logical inferences, adding knowledge to the ontology, applying rules, checking the ontology's consistency and answering queries.

The list of available OWL reasoners is available at http://www.w3.org/2001/sw/wiki/OWL/Implementations. When last accessed (March 11, 2014), it contained 20 reasoners supporting various description logic.

In this work, we utilised the reasoner Pellet, because it is suited for reasoning over OWL 2 DL-ontologies, supports DL-safe rules and has API to be called up from the Manchester OWL-API which, in turn, allows for the creation of ontology-based applications in Java.

2.3.9 Ontology Engineering Tool Protégé

A significant number of ontology editors have been designed [7, 30]. The list of up-to-date ontology editors, APIs and development environments is available at http://www.w3.org/2001/sw/wiki/OWL/Implementations. Currently (as accessed on March 11, 2014), it contains five positions heqadedby the editing tool Prot?g?. We chose this environment for the creation of the ontology and OWLAPi to process ontologies in Java.

Protégé is a free, open-source platform that provides a suite of tools designed to modify and operate with ontologies [12, 92]. The ontologies can be stored in different formats (text standard, in JDBC, UML, XML, RDF, OWL, OWL 2). Protégé contains many plug-ins, which are used to modify and process ontologies. Plug-ins enable the creation of axioms, the visualization of the ontology, the addition of SWRL-rules and reasoning (for example, Pellet) [50]. Protégé is constantly being developed and improved.

2.3.10 Ontology Alignment

Many terms are used to express different types of operations over ontologies [9, 103]. It is important to note that, although some of them sound synonymous, they have different meanings. Moreover, different authors may use the same terminology to identify slightly different notions. In order to avoid ambiguity, we will give a short overview of the related terminology.

The most frequent case is confusing ontology mapping [57], matching and alignment [29,63]. In [39] and [42], ontology mapping is defined as assigning functions across the entities of one ontology in such a way that they correspond to some entities in a second ontology. These functions are expressed as a set of additional axioms, which are stored in a separate file. Ontology mapping is oriented, as the functions can be used to perform mapping only in one direction.

Ontology matching is the process of finding corresponding entities between two ontologies, where the set of possible relations is fixed. A correspondence between entities e and f of ontologies O_1 and O_2 is defined as a quintuple.

$$correspondence = (id, e, f, r, n) \tag{2.7}$$

In equation 2.7, id is the identifier of correspondence in a set of alignments, r is the type of relationship between the entities, such as equivalence and subsumption, ($r \in R$) and n is a real number between zero and one ($n \in [0, 1]$). The number n reflects the level of confidence that the relation r between the entities e and f will hold.

Euzenat and Shvaiko [42] define alignment as the set of correspondences between the ontologies' entities. In other words, according to [42], matching is the process and alignment is the result thereof.

Ehrig uses a different approach to separate alignment from matching. He outlines the difference between matching and alignment by restricting the set of possible relations R to equivalence for only the latter case. He includes the resulting sets of correspondences in both definitions. Thus, for the task of ontology alignment, we can use quadruples instead of quintuples. Informally, Ehrig defines ontology alignment as finding the ontologies' entities that have the same meaning.

The main application of ontology matching and alignment is overcoming semantic heterogeneity [40, 59].

In this work, we will adopt Ehrig's definition of ontology alignment and use it in the sense of finding pairs of semantically equal ontology entities.

2.3.11 Similarity Measures

Similarity measures express similarity between two objects. As applied to ontologies, these may be any ontology entities, including instances, classes or relations.

According to Euzenat and Shvaiko [42, p.73], a similarity measure is a function of a pair of entries to a real number. It satisfies the following properties, where $\sigma(x, y)$ stands for the similarity function and o means ontology.

1. Positiveness

$$\forall x, y \in o, \sigma(x, y) \geq 0$$

2. Maximality

$$\forall x \in o, \forall y, z \in o, \sigma(x, x) \geq \sigma(y, z)$$

3. Symmetry

$$\forall x, y \in o, \sigma(x, y) = \sigma(y, x)$$

WordNet-based Similarity Measures

According to [84,95], WordNet is a lightweight lexical ontology. It contains various concepts that are connected to each other by certain types of relations [85, 86]. The idea of *synset* is the basis of WordNet. A synset is a set of words that are synonyms (have approximately the same meaning). Each concept has a gloss, which describes the idea in the natural language. WordNet provides hypernyms and hyponyms (super concept/ sub concept), metonyms (part of) and various other types of relationship between the words. Euzenat and Shvaiko described WordNet as "a partially ordered synonym resource" [42, p. 87]. Many similarity measures are based on the relationships between the concepts in WordNet [31,32].

Synonymy

Synonymy similarity is a simple measure based on the relationship of synonymy between the concepts. It is computed in the following way [42, p. 88].

$$sim_{syn}(S1, S2) = \begin{cases} 1 & \text{if } \sum(S1) \bigcap \sum(S2) \neq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2.8}$$

Here:

- $\sum(S1)$ is the synonym resource of the first concept.

- $\sum(S2)$ is the synonym resource of the second concept.

Example:

1. sim_{syn} (Data, Information) =1, because these concepts belong to one synset in WordNet 2.1.
2. sim_{syn} (Data, Model) =0, because these concepts are not synonyms and do not share any synonyms.
3. sim_{syn} (Management, Instruction) =1, because despite these concepts belong to different synsets in WordNet 2.1, they both share the synonym " Direction".

Wu-Palmer

This similarity measure was named after its creators, Wu and Palmer [117]. It utilises hypernym/ hyponym relationships in WordNet [42, p. 89]. These relationships reflect possible generalisations of the concept. The Wu-Palmer measure is calculated in accordance with the following formula. It counts similarity over a hierarchy $H = \langle O, \leq \rangle$

$$sim_{wp}(S1, S2) = \frac{2 \times \delta(S1 \wedge S2, \rho)}{\delta(S1, S1 \wedge S2) + \delta(S2, S1 \wedge S2) + 2 \times \delta(S1 \wedge S2, \rho)} \tag{2.9}$$

Here:

- ρ is the root of the hierarchy.
- $\delta(C1, C2)$ is the number of intermediate edges between the concept C1 and another concept, C2.
- $S1 \wedge S2 = S3 \in O; S1 \leq S3 \wedge S2 \leq S3$.

Example

1. sim_{wp}(Data, Information);

   S1 = Data;

   S2 = Information.

   These words are synonyms. For this reason, $\delta(S1, S1 \wedge S2) = \delta(S2, S1 \wedge S2) = 0$. Thus sim_{wp}(Data, Information) = 1.

2. sim_{wp} (Database, Camomile);

   S1 = Database;

   S2=Camomile.

   These words have only one common hypernym, which is exactly the root of WordNet (this concept is ”Entity”). For this reason $\delta(S1 \wedge S2, \rho) = 0$. Thus sim_{wp} (Database, Camomile) = 0.

3. sim_{wp} (Database, Database Management System);

   S1 = Database;

   S2 = Database Management System.

   These concepts have the nearest common hypernym ”Communication”. This concept has two more hypernyms before the root. They are ”Abstraction” and ”Abstract entity”. Thus, $\delta(S1 \wedge S2, \rho) = 3$. There are two more hypernyms between ”Database” and ”Communication”. These are ”Info” and ”Message”. Thus $\delta(S1, S1 \wedge S2) = 3$. There are five hypernyms between ”Database Management System” and ”Communication”. These are ”Software”, ”Code”, ”Coding system”, ”Writing” and ”Written communication”. Thus $\delta(S2, S1 \wedge S2) = 6$.

   On balance, $sim_{wp}(Database, DatabaseManagementSystem) = \frac{2\times3}{3+6+2\times3} = 0.40$.

This similarity measure is sensitive to the shortest path in the hierarchy of concepts [19].

This similarity measure takes into account the fact that two concepts near the root of a hierarchy are close to each other in terms of edges but can be very different conceptually, while two concepts that fall under one hierarchy but which are separated by a larger number of edges should be closer semantically.

Edit Distances

Levenshtein Edit Distance

Edit distances include the Levenshtein, Needleman-Wunsch and various other metrics. They reflect a dissimilarity between the concepts. According to Euzenat and Shvaiko, [42, p.73], a dissimilarity measure is a function δ from a pair of entries to a real number expressing dissimilarity between them. It satisfies the following properties.

1. Positiveness

$$\forall x, y \in o, \delta(x, y) \geq 0$$

2. Minimality

$$\forall x \in o, \delta(x, x) = 0$$

3. Symmetry

$$\forall x, y \in o, \delta(x, y) = \delta(y, x)$$

The Levenshtein distance is used to count the number of insertions, substitutions and deletions needed to transform one string into another. The Needleman-Wunsch algorithm is the Levenshtein distance with a higher value assigned to operations of insertion and deletion. These characteristics are commonly used to compute similarity between strings that may contain spelling mistakes.

The similarity measure based on the Levenshtein distance is computed as $sim_{Lev} = 1 - dist_{Lev}$.

Example

1. $sim_{Lev}(Databse, Database)$.

   The longer word contains eight letters. To convert string ”Databse” to ”Database”, we need to insert the letter ”a” between ”b” and”s” in the first word. Thus, $dist_{Lev} = \frac{1}{8} = 0.125$ and corresponds to $sim_{Lev} = 1 - 0.125 = 0.875$.

2. $sim_{Lev}(Data, Database)$.

   The longer word contains eight letters. To convert the string "Data" to "Database", we need to add four letters to the first word. Thus, $dist_{Lev} = \frac{1}{8} \times 4 = 0.5$ and corresponds to$sim_{Lev} = 1 - 0.5 = 0.5$.

This measure is quite useful for comparison of the strings, because it helps to overcome expected spelling mistakes and may also be useful for finding words with similar roots (see Example 2), which are extremely likely to be semantically similar.

Soundex and Jaro-Winkler

Soundex and Jaro-Winkler is another measure that enables the similarity computation of misspelled words.

It utilises a phonetic algorithm, Soundex, which encodes the letters according to how they are pronounced. The algorithm consists of the following steps [4].

1. Retain the first letter in the string and delete all other occurrences of the letters a, e, i, o, u, y, h and w.
2. Replace consonants with digits in the following way:

   $b, f, p, v \rightarrow 1$

   $c, g, j, k, q, s, x, z \rightarrow 2$

   $d, t \rightarrow 3$

   $l \rightarrow 4$

   $m, n \rightarrow 5$

   $r \rightarrow 6$
3. If two or more letters with the same number are adjacent in the original name (before step 1), remove all letters except for the first. If two letters with the same number are separated by "h" or "w", they are coded as a single number. If such letters are separated by a vowel, they are coded twice. This rule is also applicable to the first letter.

4. Iterate the previous step until the sequence consisting of one letter followed by three numbers remains. If there are not enough letters in the words, they should be appended with zeros. If more than three letters are left, all the extra numbers are deleted.

Once the Soundex encodings of the strings are received, they are compared with the help of the Jaro-Winkler algorithm.

The similarity measure Jaro is computed in the following way [42, p. 80].

$$sim_{Jaro} = \frac{1}{3} \times (\frac{|com(S1,S2)|}{|S1|} + \frac{|com(S2,S1)|}{|S2|} + \frac{|com(S1,S2)| - |transp(S1,S2)|}{|com(S1,S2)|}) \tag{2.10}$$

where $S[i] \in$ com (S1,S2) if and only if

$\exists j \in [i - (min|S1|,|S2|)/2i + (min|S1|,|S2|)/2]$

Here:

- $|S1|$ and $|S2|$ are the lengths of compared strings.
- com (S1,S2) and com (S2,S1) are the common symbols in the strings S1 and S2 that were found in compliance with the condition.
- transp(S1,S2) are the elements of com(S1,S2) that occur in S2 in a different order.

The Jaro-Winkler similarity measure is an improved version of Jaro.

$$sim_{JaroW} = sim_{Jaro} + P \times \frac{1 - sim_{Jaro}}{10} \tag{2.11}$$

Here:

- sim_{Jaro} is the Jaro similarity computed in accordance with the Equation 2.10.
- P is the number of symbols in common prefixes.

The amount of numbers is then reduced to one starting letter followed by three digits.

Example

1. $sim_{SoundexJW}$ (Databse,Database).

   First, we converted both words to Soundex strings and obtained the following.

   Databse = D312 (D a is deleted, 3 for t, 1 for b, 2 for s, e is deleted).

   Database = D312 (D, a is deleted, 3 for t, 1 for b, a is deleted, 2 for s, e is deleted).

   Thus, we see that the two Soundex strings are identical and $sim_{SoundexJW}$ (Databse,Database)=1.

2. $sim_{SoundexJW}$ (Data,Database)

   First, we converted both words to Soundex strings and obtained the following.

   Data = D300 (D, a is deleted, 3 for t, a is deleted, append two zeros).

   Database = D312 (D, a is deleted, 3 for t, 1 for b, a is deleted, 2 for s, e is deleted).

   We then computed sim_{Jaro}(Data, Database) = $\frac{1}{3} \times (\frac{2}{4} + \frac{2}{4} + \frac{2-0}{2}) = 0.67$.

   Thereafter, we calculated sim_{JaroW} (Data, Database) =$0.67 + 2 \times \frac{(1-0.67)}{10} = 0.73$.

   Thus, $sim_{SoundexJW}$ (Data,Database)=0.73.

Information Content Measures

Examples for the following measures are not provided because they depend on the text corpus utilised.

Resnik

This measure is based on the hypothesis that the more information the two concepts share, the more similar they are. The shared information is reflected by the information content of their nearest common subsumer [19, 100].

$$sim_{Res}(S1, S2) = max_{c \in S(S1,S2)} IC(c) \tag{2.12}$$

Here:

- $S(S1, S2)$ is the set of concepts, that subsume S_1 and S_2.
- $IC(c) = -logp(c)$. c is a concept. $p(c)$ is the probability of encountering c in a given corpus.

The weak point of this measure is it being equal to $IC(c1)$, not 1, if the concepts are equal. The second drawback is that any two concepts with the same most specific common abstraction have the same similarity value.

Lin

According to Lin [76], the similarity between S1 and S2 is measured by the ratio between the amount of information needed to state the commonality of S1 and S2 and the information needed to fully describe what S1 and S2 are [76]. Lin is an elaborate variant of Resnik's similarity measure. It is computed in the following way.

$$sim_{Lin} = \frac{2 \times sim_{res}(S1, S2)}{IC(S1) + IC(S2)} \tag{2.13}$$

The advantage of Lin, when compared to Resnik, is that it considers not only the information shared by two concepts, but also the data that differs. The Wu-Palmer similarity measure is a particular case of Lin [19].

DISCO

DISCO (extracting DIstributionally related words using CO-occurrences) is an elaborate version of the Lin similarity metric [70]. The authors distinguish between the notions of semantic and distributional similarity. They emphasise that semantic similarity deals with concepts, taking into account the word senses. At the same time, distributional similarity is more concerned with the comparison of words encountered in the contexts, independent of their (possibly) different meanings. The creators of the measure also discuss the difference between the concepts of distributional relatedness and distributional similarity. They note that

the first concept operates using the bags of words, while the second considers the co-occurring words in the same syntactic relationships. They position DISCO between the categories of distributional similarity and relatedness, due to its algorithm.

There are two types of DISCO metrics. These are DISCO1 and DISCO2 [70, 71]. DISCO1 reflects the first-order similarity and relatedness between words and is calculated based on collocation sets of the words. DISCO2 is a measure that characterises the relatedness and similarity between the terms based on their sets of distributionally similar words. It compares the second order vectors of two words. The second order vector for a word contains the words that occur together with it, as well as those that occur in similar contexts.

In order to calculate any of the DISCO measures, one must load the language data packet according to which it is computed. The archive already contains the pre-computed database of collocations and distributionally similar words. This index speeds up the computation of the measure. Currently, there are packages that support nine languages, including English and Russian. According to [70], DISCO shows a higher correlation with semantic relatedness judgements by humans than do the WordNet-based similarity measures.

2.3.12 The General Alignment Process

Despite the wide variety of ontology alignment algorithms that exist nowadays, most of them share the same ideas regarding the main steps. Ehrig outlines the six main stages that comprise an average alignment algorithm ([39, p.62]).

An ontology alignment algorithm receives two (or more) ontologies as input.

The first step is called *Feature Engineering*. The key point at this stage is to excerpt all the possibly useful knowledge regarding each of the ontology entities, while simultaneously avoiding all extraneous information. For example, it may be sensible to excerpt the class to which an individual belongs, its data type properties and its relationship to values, while it may be sensible to exclude the set of super- and subclasses, data type and object properties for a class.

The second step is called *Search Step Selection*. It dictates the rules regarding the choice of the ontology entities' pairs for alignment. For example, the algorithm may align only classes with classes and not with relationships.

The third step is called *Similarity Computation*. This step details the similarity measures that should be used for each type of ontology entities' pair from step 2. For example, it may be sensible to calculate semantic similarities between the words used as class labels.

Similarity Aggregation is the next stage of an alignment algorithm. The choice of an appropriate aggregation strategy may depend on the task. The use of triangular conorms for this purpose is quite common. Normally, an alignment algorithm will combine several classes of similarity measures, such as semantic, structural and so on. In turn, each class may include various measures. In this case, the measures should first be aggregated inside each of the classes. It is sensible to calculate the average value at this step, as the matches will be combined afterwards. In this case, the weighted average strategy with different weights set for the classes of measures may be preferable, depending on the importance of their contribution to the alignment task. The weights may be set as default or tuned to find the values that give the best results.

Once all the similarities have been computed and aggregated for each pair of ontological entities, *Interpretation* takes place. The aim at this stage is to select the values of the aggregated similarities between the pairs of entities that count as satisfying the chosen type of relationship. The usual approach to this task is to assign alignments based on a threshold. The idea behind this solution is the assertion that a relation holds between two entities if the aggregated similarity value is more (or less) than a certain value. For example, if the similarity value between entities e_1 and e_2 is more than 0.8, then they are equal.

The last stage is *Iteration*, which, in reality, includes the repetition of all the previous steps. However, it is important to note that, usually, all repetitions should include fewer computations than the first run, because they consider the results of the previous iterations. Normally, an ontology alignment algorithm includes several types of matchers, such as semantic and structural. The measures that calculate similarity values based on the natural language words representing the ontologies' entities may be computed once. However, calculation of the similarity between two entities as the structural units of a graph may differ, because the similarity value of one pair may depend on the similarity between two other entities. For this reason, the structural similarity may be computed more than once. The number of iterations maybe fixed or may be restricted by a fixed time constraint.

2.3.13 The Ontology Alignment Evaluation Initiative

As a result of the growing need to gather, compare, qualify and announce the best practices in the field of ontology matching and alignment, the Ontology Alignment Evaluation Initiative (OAEI) evolved. On a yearly basis, the OAEI announces a call for papers and organises a workshop to welcome the researchers contributing to the field. The OAEI has designed five groups of datasets. Each of them allows for the evaluation of the algorithms based on the class of matching tasks they should solve. The participants may compete in any number of the categories.

The first dataset is called the "Benchmark" and contains one ontology, which is systematically altered in such a way that enables the detection of the strengths and weaknesses of the alignment algorithms. The second dataset contains two expressive ontologies, anatomy and conference, each of which consists of approximately 3000 classes. The task is to find as many correspondences as possible. The third dataset presents a challenge to algorithms performing the oriented matching. The dataset contains ontologies, representing the catalogues of educational courses of two universities. The aim of oriented matching is to recognise not only equivalence, but also subsumption relationships between the ontologies' entities. The fourth dataset deals with model matching. It aims at comparing model matchers to ontology matchers. The last dataset evaluates how the participants cope with instance matching. The track changes from year to year, as well as according to the nature of the task. For example, in 2011, the participants were obliged to rebuild the links in the ontology of the New York Times and to match it to three other ontologies.

The participants present their algorithms, the results of alignment in a specialised format and papers describing the approach. For some datasets, the results are compared to pre-defined reference alignments so that the researchers have idea of how well they performed while, in other cases, the procedure is blind. The organisers prepare and present a comprehensive report, choosing the best algorithms for each task.

By preparing the yearly reports that compare current ontology matching and alignment algorithms, the Ontology Alignment Evaluation Initiative enables the sharing and spreading of best practices, further development of ideas and, finally,

documents the state-of-the-art in this research area. Later, we will present an overview of the ontology alignment algorithms that proved to be the best, based on their performance at OAEI-2011.

2.3.14 Existing Ontology Alignment Approaches and Algorithms

ASMOV

This algorithm was the top contestant in OAEI 2007-2010, and did not participate in 2011. However, it outperformed all of the competitors in 2011, according to the Benchmark dataset. In 2009, ASMOV showed better results than did the other four systems competing in the oriented matching category. OAEI encouraged the researchers to challenge the oriented matching track in 2011 as well, but they did not achieve a good response. Thus, we decided to provide an overview of ASMOV as one of the best ontology alignment systems designed so far.

The abbreviation ASMOV stands for the Automated Semantic Mapping of Ontologies with Validation. It was designed to facilitate the integration of various data sources presented in the form of ontologies. The algorithm computes mappings between classes, objects and data type properties, and individuals.

ASMOV uses four features for similarity computation between the ontologies' entities. According to [119, p.152], these are lexical elements, relational structures, internal structures and extensions.

The pre-processing stage performs a calculation of the similarities between lexical elements, including identifiers, labels and comments. The second stage includes similarity calculation. The matcher computes similarities between the relational structures of ontologies, addressing them as ancestor-descendant hierarchies. The internal structure similarity takes into account restrictions on properties, the properties' types, domains and ranges, and data values. Extension is the third stage, which computes similarities between the individuals in classes and their property values. The pre-alignment stage aggregates the similarity values of the four types, using the weighted average strategy.

The semantic verification of the alignments is the essence of ASMOV. It clears the resulting alignment of semantically inconsistent mappings and the causes thereof.

ASMOV outlines five types of inconsistencies, including many-to-many entities correspondences, the disjointedness-subsumption contradiction (if two classes are disjoint in one ontology, they cannot be aligned with parent and child classes in another) , crisscross correspondences (a parent-child class of the first ontologies cannot be aligned with child-parent classes of the second ontology), subsumption incompleteness (parent and child classes of one ontology are mapped to two classes of the second ontology, if the subsumption relation holds for the latter two), and domain and range incompleteness (two properties can be aligned if, and only if, their domains and ranges are mapped correspondingly).

AgreementMaker

The developers describe AgreementMaker as a system with "a powerful user interface, a flexible and extensible architecture, an integrated evaluation engine that relies on inherent quality measures, and semi-automatic and automatic methods."([33, p.114]). It has successfully participated in the OAEI since 2009.

The AgreementMaker includes different matchers, as does any other ontology alignment system. The automatic choice and combination of the different techniques is the core specialty of the system. The profile of each ontology is created before matching. It contains such features as relationship and inheritance richness, for example. The values of these properties enable the automatic configuration of the similarity matchers.

CODI

Combinational Optimisation for Data Integration (CODI) is another successful participant in OAEI-2011. The system produces one-to-one alignments between classes, individuals, and the object and data type properties of the ontologies. According to [58, p.134], CODI utilises the syntax and semantics of Markov logic, which combines first-order logic with undirected, probabilistic graphical models. A Markov logic network is a set of first-order formulae with weights, in which the greater the value of the weight, the higher the probability of the formula's correctness.

As with other alignment algorithms, the CODI implements several similarity matchers, whose values are aggregated using average, maximum or weighted

strategies. CODI utilises such string similarity measures as the cosine, Levenshtein, Jaro-Winkler, Smith-Waterman-Gotoh, the overlap coefficient and Jaccard. The CODI has a special algorithm for the alignment of instances. According to the strategy, the structures of ontologies are matched at the first stage. Thus, before searching for like instances, the CODI supposes that it processes the same TBoxes in terms of the description logic knowledge base. Thereafter, the alignment of the instances commences. The aim of the first step is to compute basic anchor-alignments. As can be understood from the paper, this means that the individuals of a randomly chosen class of the first ontology are extracted and are matched with the individuals of the corresponding class in the second ontology. All the alignments with a similarity value above a certain threshold are considered to be in the anchor-alignment. The similarities are then computed for all the individuals linked to the members of the first anchor alignment by the corresponding object properties. The process continues until no anchor alignments can be produced.

YAM++

The founders of YAM++ call their system "a (not) yet another matcher" and propagate it as being a flexible, self-configurating and extensible ontology matching system [90, p. 228]. The YAM++ system uses machine-learning techniques for the combination of similarity measures, and applies a similarity propagation algorithm for the detection of other possible, existing alignments. Firstly, the terminological similarities of classes are calculated using such string-based metrics as Levenshtein, Smith -Waterman, Jaro, Jaro-Winkler and Monge - Eklan, and language-based metrics such as Lin, Jing - Conrath, and Wu and Palmer. Secondly, the similarities for the instances of the classes are computed. The results of these steps are combined to achieve element-level similarity. The next step is called similarity flooding, and is used to compute structural similarity for the ontologies' entities. Final mappings are introduced by the weighted aggregation of the element and the structural-level similarities. Further on, the algorithm removes inconsistent alignments.

Cluster-based Similarity Aggregation

The creators of this algorithm posit it to be "an automatic similarity aggregating system for ontology matching" [110, p. 142]. The current system performs one-to-one ontology alignments between concepts, data types and object properties.

The Cluster-based Similarity Aggregation (CSA) uses five types of similarity metrics, including the string-edit distance, Wu and Palmer and the profile (based on the surroundings of an entity, such as individuals or subclasses for comparing two concepts). Thereafter, the similarities are aggregated using the weight average strategy. Weight estimation is the essence of the CSA. The aim of the process is to define the weight of each similarity measure. The main idea is to divide all alignments into two categories - matching and non-matching pairs. The CSA defines a similarity metric as being effective if it satisfies two criteria, namely distinguishing matching pairs from non-matching pairs, and the number of matching pairs that seek the number of entities in a smaller ontology. The K-means algorithm is used to cluster alignments into two classes, as explained above. The cluster with the higher mean represents the matching pairs. The values from the non-matching set are filtered out of the similarity matrix. The weight for this similarity measure is estimated as the ratio of the number of the non-empty rows in the filtered similarity matrix over the number of values in the matching set. The number of rows represents the number of one-to-one possible alignments. The structural similarity is calculated for the classes only. The pre-alignment is extracted using the stable marriage problem algorithm, wherein the sets of the entities of two ontologies represent men and women, while the similarity values are used to build the lists of priorities for each person (or ontological entities, in this case).

MapSSS

The algorithm MapSSS is aimed at using simple similarity measures as opposed to complicated alignment techniques [28, p.184]. The triple 'S' in the title of the method stands for syntactical, structural and semantic measures that are utilised by the system. The algorithm treats ontologies as directed graphs with OWL classes, individuals and data type properties representing the nodes, while the relations are converted to the edges. The system produces one-to-one alignments.

Currently, the algorithm utilises only the first two types of similarity measures announced in the title. The Levenshtein distance is used for computing the syntactic similarity. This structural metric treats ontologies as simple graphs with the following restrictions applied. Firstly, only entities of the same type can be matched. Secondly, the edges can be compared only if they link nodes of the

same type. The third rule proclaims that, if an ontological entity is a neighbour to another entity that has already been mapped, then it can only be matched to the neighbour of the entity present in the existent mapping.

2.4 Summary

The main objectives of this chapter were to study current approaches, materials and criteria used for the comparison of educational courses and modules, and to justify the research topic in the real world. The comparison of courses is timely for such tasks as prior learning and degree recognition, as well as for joint educational programme design, which encourages collaboration between international higher education providers and facilitates mobility for students and job applicants.

The research showed that, currently, the comparison of the courses is performed manually by the groups of experts in the field of education. It is obvious that a human cannot be completely replaced by a computational algorithm for prior learning or degree recognition. However, partial automation could seriously reduce the involvement of highly qualified experts in the initial stages of the process and could provide additional and visual information in the latter stages of the decision-making process.

The shift from teacher- to student-centred education is a modern tendency in the sphere of higher education. The critical analysis of the current situation in the European Higher Education Area and documents produced by the Bologna Working Group outlined the education cycles, profiles, ECTS credits and learning outcomes and competencies as the cornerstones for the introduction of a system of comparable and compatible HE degrees. The educational courses and modules can be compared only if they belong to one education level and have the same amount of ECTS credits. Classification of educational programmes in accordance with profiles is too vague, because they are very general and are comprised of a considerable number of fields of education that are more specific.

The study of the documenting of educational courses and modules in Russia and the United Kingdom showed that the courses are described at a high level of abstraction. Programme specifications provide an overall impression of the courses, but are too general to be utilised for a high quality comparison of the

contents thereof. At the same time, the module templates contain detailed information, including the sections with the data domain's keywords and the exact learning outcomes. Unlike course competencies, the modules' learning outcomes are extremely precise and are closely related to the field of studies.

For these reasons, and taking the Bologna Recommendations into account, we decided to create an algorithm for the comparison of the educational modules based on the similarities of their keywords and learning outcomes. We also concluded that it would be scientifically valid to compare the educational courses based on the contents and outcomes of the modules of which they consist. Thus, our method for the comparison of the courses treats them as sets of their modules.

In this chapter, we presented a critical review of current knowledge representation models, including logical models, propositional and predicate calculi, production, frame and network models. We came to the conclusion that ontology belongs to the class of combined models, because it integrates the characteristics of network and of either frame or logic models. Due to these peculiarities, ontologies inherit the advantages of both approaches. The ontologies that combine semantic networks and mathematical logic are formal and semantically expressive at the same time. For this reason, we decided to use such an ontology to formally represent the educational courses and modules.

We reviewed informal and formal ontology definitions and chose the formal ontology definition. We also reviewed the basics of description logics in relation to the construction of ontologies and knowledge bases. We chose OWL 2 DL as the representation language for our ontology because it is expressive and decidable. We paid special attention to the Semantic Web Rule Language, which allows for the expansion of the descriptions logics-based ontologies via the inference rules. The use thereof can profoundly simplify the ontology population with instances and could improve its semantic expressiveness. We decided to use the Protégé ontology engineering tool, because it supports OWL 2 DL, allows for reasoning, contains the plugin to add SWRL-rules, has a comfortable user interface and can be downloaded and utilised for free.

We then discussed the concept of ontology alignment, having presented the typical structure of any matching algorithm. The algorithms that performed best at the recent Ontology Alignment Evaluation Initiative were described. We also

analysed the most common similarity measures used by ontology matching techniques. We decided to create a specialised ontology alignment algorithm for the comparison of educational courses and modules, because this technique is suited to finding correspondences between entities. Ontologies store and formally represent information regarding the concepts, their properties and the interrelation of the programme specifications and module templates. The ontology alignment algorithm finds similarities based on all of the knowledge stored in this formal representation.

Chapter 3

Ontology of an Educational Course and Modules

3.1 Introduction

The main objective of this chapter is to design an ontology, which would allow to store the information necessary for comparison of educational courses and modules. Another aim is to propose the method for partial automation of ontology population from the natural language documents.

To achieve our goal we break the problem into several tasks. At first we outline, which information is to be modelled. We assume that the ontology touches the three subdomains: course and module with their interrelations, learning outcomes and the modules' data domains. In order to define the notions to be reflected in the ontology we build mathematical models of each of them. Afterwards we convert the formal representations to ontologies and describe in detail the classes, data and object properties, and restrictions set over them. We also create grammars, which allow markup of the "Keywords" and "Learning Outcomes" sections of the module templates. The tagged text can be utilized to automate ontology population.

3.2 Methodology for Creation of an Ontology of an Educational Course and Module

The methodology for creation of the ontology of an educational course and module contains two main stages: building and population of the ontology.

3.2.1 Ontology Building

The methodology contains the following main steps.

1. Decompose the task into building several smaller models, the combination of which gives the full model of the educational course, module and their data domain. We outlined the course and module, a learning outcome, and data domain of a module as the subdomains, which can be modelled almost independently on each other and linked together afterwards.
2. Build a formal model of an educational course and module based on the set theory.
   (a) Outline the key concepts of an educational course and module and their characteristics, which can be extracted from programme specifications and module templates and used for their comparison.
   (b) Divide the concepts into the sets based on their mission.
   (c) Represent each concept as a tuple, where constituents should reflect the attributes of the concepts and possible relations between them as functions.
3. Convert the mathematical model to the set of ontologies.
   (a) Convert the sets of the mathematical model to the classes and datatype properties of the ontology.
   (b) Convert the functions to object properties of the ontology.
   (c) Assign the restrictions on the properties.
4. Produce additional relations and restrictions linking the smaller ontologies into one ontology.

3.2.2 Ontology Population

Population of the ontology is a separate task, which consists of the following steps.

1. Populate the basic sub-ontology of a course and modules.
2. Populate the sub-ontology of the data domains of modules.
3. Populate the sub-ontology of the learning outcomes.

In this work we introduce a semi-automated methodology for ontology population. The key points, which should allow automation, are the following. The first step is preparation of the ”short module templates”, which contain only the sections with information, which should be extracted to the ontology. The second step is paying special attention to the keywords and the learning outcomes, as they are the main entities contributing to the comparison of the educational modules and thus courses. We address the ”Keywords” section in a template as a set of named topics, each of which contains a number of keywords. For the sake of automation of ontology population we designed the formal grammars of a keyword (GK) and of a learning outcome (GLO), which allow automatic annotating of the pieces of text to be extracted into certain classes of the ontology. The GK and GLO are presented in detail lower in this chapter. The third step is identifying the cases, where the classes and properties can be populated with the help of logical inference. In our work we also created a number of SWRL-rules, which enable automatic population when fired by a reasoner. They are described in detail in the Chapter ”Implementation”.

3.3 The Basic Ontology of an Educational Course and Module

3.3.1 The Formal Model Of an Educational Course and Module

We ground our model on the concept of a programme specification, which is the main document, describing the educational programme and its contents. For this

reason we address a programme specification as the cornerstone of the system and represent it as the following quadruple *PS*.

$$PS = (EdCProperties, MTs, EdCs, Competences) \tag{3.1}$$

In the Formula 3.1 the *EdCProperties* is the set of the properties of an educational course, which are the following.

$$EdCProperties = (PSTitle, Level, Profile, PSCredits) \tag{3.2}$$

The Quadruple 3.2 contains the following constituents.

1. *PSTitle* is the natural language title of the programme specification.
2. *Level* contains one value from the set *Level*, which includes possible types of the educational levels. $Level = \{Undergraduate \mid Postgraduate\}$.
3. *Profile* stands for the field of studies, to which the programme specification belongs (e.g. computer science).
4. *PSCredits* is the number of credits, which the educational course costs.

Any programme specification contains the references to the educational modules, which a student studies as a part of it.

The critical review of the British and Russian programme specifications showed, that a module can be included into the document in several ways. The situation is illustrated in the picture below.

The Figure 3.1 shows, that three main types of a module's inclusion into a programme specification exist. The first of them are compulsory disciplines, which a student must take in order to finish the programme. The optional disciplines form the second type: a student may take them or may not depending on his wish. The third type is called elective disciplines. In this case a group of optional modules is introduced. A student should choose from one to several disciplines from the set in accordance with some rules such as "choose any two modules from the set of four". Practically, once the choice of the particular modules is done, the elective disciplines become compulsory.

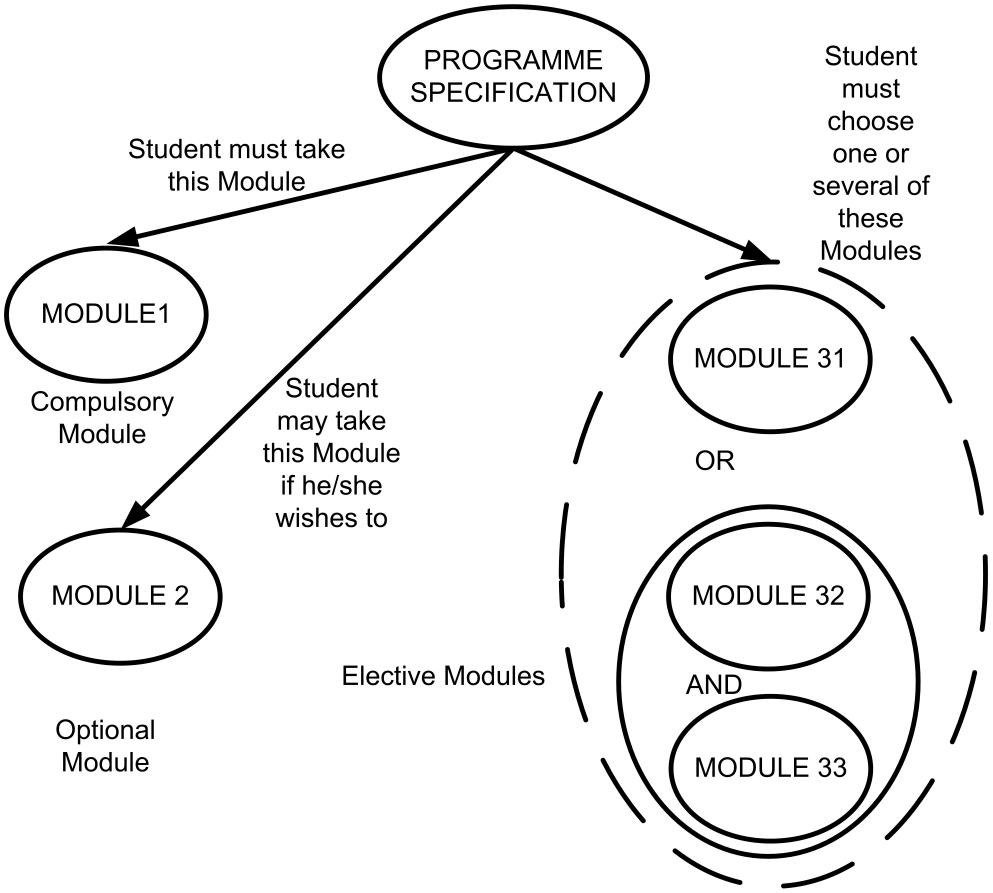

Figure 3.1: The Types of Modules' Inclusion in a Programme Specification

The *MTs* in the Definition 3.1 stand for the set of the module templates, which are mentioned in the programme specification.

$$MTs = (CompMTs, OptMTs, ElMTs). \tag{3.3}$$

As can be seen from the Equation 3.3, all the module templates in a programme specification are divided into the three non-intersecting sets of the templates: compulsory modules (*CompMTs*), optional (*OptMTs*), and elective modules (*ElMTs*).

$$\text{ElMTs} = (\text{ElMT, Rule}) \tag{3.4}$$

The Equation 3.4 defines the group of elective modules *ElMTs*. It contains two constituents: *ElMT*, which is the set of module templates of the elective modules,

and *Rule*, which is the rule in accordance with which the modules from the *ElMTs* are to be elected. Each *Rule* is characterized by two parameters as follows.

$$\text{Rule} = (\text{RuleDef}, \text{ParVal}) \tag{3.5}$$

In the Equation 3.5 the following notation is utilized.

1. *RuleDef* is the natural-language definition of the rule. For example, choose any two modules from the group.
2. *ParVal* is the numeric value of the parameter used to define the rule. In the above example ParVal would equal the value of two.

Each module template, either compulsory, optional or elective, contains the following information about an educational module, presented as quadruple.

$$\text{MT}_i = (MTitle, MCredit, DD, LO) \tag{3.6}$$

The following denotations are used in the Definition 3.6.

1. *MTitle* contains the natural-language title of a module.
2. *MCredit* indicates how many educational credits the module costs.
3. *DD* is the URI of an ontology, describing the data domain of an educational module. This structure is depicted in detail in the following section.
4. $LOs = \{LO_1, \ldots, LO_j, \ldots, LO_m\}$ is the set of the learning outcomes LO_j of an educational module.

In the Definition 3.1 the *EdCs* stands for the set of educational courses, which can be designed based on the programme specification *PS*.

$$EdCs = \{EdC_1, \ldots, EdC_i, \ldots, EdC_n\} \tag{3.7}$$

The EdC_i from the set 3.7 is presented as the set of its educational modules.

$$EdC_i = (MT_{C_i})$$

An educational course EdC_i contains the set of the educational modules $MT_{C_i} = CompMT \bigcup OptMT \bigcup ElMT_{Rule_i}$, where $CompMT$, $OptMT$ are the sets of compulsory and optional modules respectively. $ElMT_{Rule_i} \in$ ElMTs, where $ElMT_{Rule_i}$ is the result of the i-th variant received by application of the $Rule_i$.

A programme specification is also characterized by the set of competences $Competences = \{Competence_1, \ldots, Competence_i, \ldots, Competence_m\}$, where each competence is represented as the following triple.

$$Competence_i = (IdC, CompCategory, CompDescr) \tag{3.8}$$

1. *IdC* is the identifier of a competence.
2. *CompCategory* is the name of the class, to which the competence belongs (e.g. general cultural, professional etc.).
3. *CompDescr* is the natural language sentence, describing the competence.

3.3.2 Creation of the Ontology based on the Formal Model of an Educational Course and Module

The basic ontology of an educational course and module was designed in accordance with the mathematical model presented in the previous section. It consists of the following classes and properties.

The Classes of the Ontology of an Educational Course and Module

The ontology contains the following classes.

1. **Programme Specification** contains the names of the programme specifications of a university. The individuals of this class have datatype properties *PSTitle* (type string) and *PSCredits* (type integer). It corresponds with the Definitions 3.1, 3.2. Additionally, we decided to add such datatype string properties as *University*, *Faculty* and *Department*, which may provide additional information about the PS.

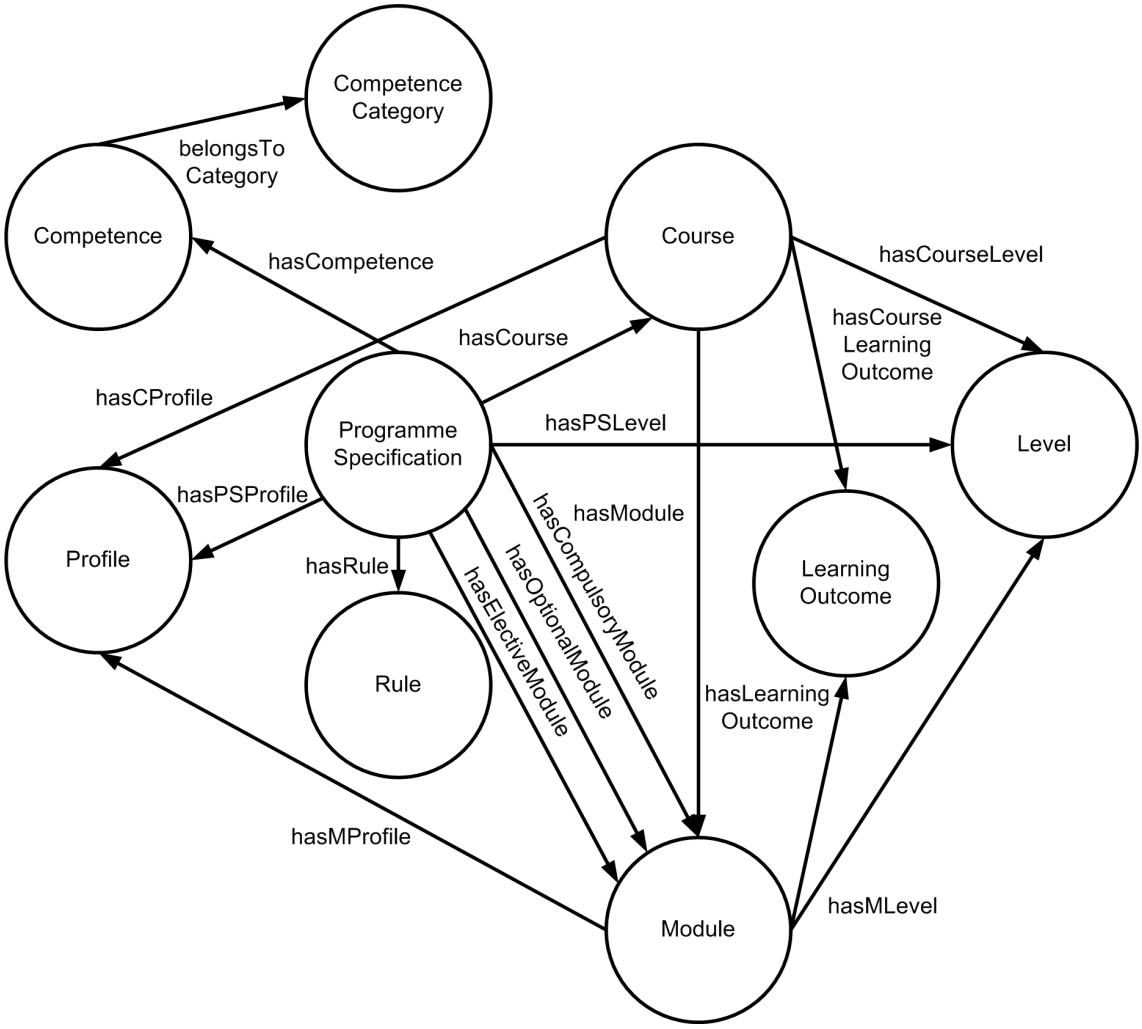

Figure 3.2: The Basic Ontology of a Generic Educational Course and Module

2. **Course** contains the educational courses, which can be received from the programme specification by choosing different combinations of optional and elective modules. It corresponds with the Definition **??**. The course also has such datatype string properties as *University*, *Faculty* and *Department*, which are inherited from its PS.

3. **Competence** is the class, which stores competences of a programme specification. Its individuals contain the datatype property *CompDesr* (type string). It is defined in the Equation 3.8 of the formal model.

4. **Competence Category** contains the titles of classes, to which a competence may belong.

5. **Module** contains the titles of educational modules, referred in the programme specification. The instances of this class have the datatype properties *MTitle* (type string), *DD* (type string) and *MCredit* (type integer). It is defined in the Equation 3.6 of the model.

6. **Rule** contains the rule, used to choose the elective modules. They correspond to the notion defined in the Equation 3.5 of the model. The individuals of this class contain datatype properties *RuleDef* (type string) and *ParVal* (type integer).

7. **Profile** contains the titles of the profiles, to which the programme specification or a module belongs. It corresponds with the Definition 3.2.

8. **Level** contains the levels, to which the programme specification belongs, e.g. bachelor. It corresponds with the Definition 3.2.

9. **Learning Outcome** contains the learning outcomes of the educational modules.

The relations between the classes of the ontology of a generic educational course and module are presented in the Appendix B.

3.4 Grammars of the Keywords and the Learning Outcomes

In order to build the ontologies of the data domain and of a learning outcome, as well as to enable their automatic enrichment from the module templates and programme specifications, the Grammars of the Keywords (GK) and the Learning Outcomes (GLO) were designed. The grammars were created based on investigation of the real ”Keywords and ”Learning Outcomes” sections from the module templates.

The construction of the grammars had the two main purposes. The first aim was to understand the syntactical and grammatical structure of the keywords and the learning outcomes and distinguish the main patterns. The results of this analysis were used to detect the entities and structure of the ontologies of the data domain and the learning outcomes. The second goal was to provide the means for automatic population of these ontologies from texts.

3.4.1 Characteristics of the Grammars

It is important to notice, that the grammars have two main assumptions.

Firstly, both reveal syntactic and only partially semantic aspects of the keywords and the learning outcomes of the educational modules. Semantics is concerned as far as it is required for recognition of the specific elements of a keyword or of a LO. We assume, that the grammars receive as an input a real list of keywords and learning outcomes. For this reason, the Grammar of the Learning Outcomes, for example, can recognise a sequence of verb and noun as an action, which is applied to some object from the data domain of the educational module. At the same time it will not be able to tell, if performing this action over this object is applicable.

Secondly, some valid combinations of the rules of the grammars result in ambiguity. This happens due to the nature of the real keywords and learning outcomes, which can have ambiguous structure like any other natural language sentences. The problem of ambiguity is solved with the help of such methods as grouping the rules, assigning priorities and purely programmatic decisions. These approaches are widely used by a variety of language processing software tools [34]. The disambiguation is discussed in more detail in the Chapter 5.

3.4.2 The Structure of the Grammars

The both grammars have the following structure.

$$Grammar = (V, N, S, R) \tag{3.9}$$

Here:

V is a set of terminal symbols. In our case V can contain any words and word sequences, which can be formed in accordance with the rules of the English language, and punctuation marks. In the grammars, the terminals are spelt in the lower-case.

N is a set of non terminal symbols, which are described in more detail below.

S is called the axiom and belongs to the set of the non terminal symbols N. The inference should always start with application of the rule, where the left hand side contains S and only S. The Grammar of the Keywords has non-terminal Topic as the axiom S. In the Grammar of the Learning Outcomes non terminal NLLOSentence is chosen as S.

R is a set of rules, which are used to recognise the keywords and the learning outcomes. From one to several rules can be applied to annotate each of the non-terminals in the grammars. For brevity the rules are accumulated in statements, where the right hand side contains all possible productions for the non-terminal from left hand side of the rule. Each single production rule inside each statement is enumerated with a number in circle (e.g. ①). The rules of the grammars are listed in the below subsections.

The following regular expression operators are used in the grammar rules.

| is a disjunction operator. It is used to accumulate the rules with the same left hand side. The disjunction operator separates expressions on the right hand side and means, that only one of them can be fired at each iteration. Here expression stands for an unbreakable sequence of symbols on the RHS, to which the LHS is to be exchanged. The symbols in each expression represent a sequential conjunction and are separated by a space.

( ) is operator "brackets". The brackets are used to bound a sequence of symbols, affected by one of Kleene operators.

* is Kleene star or an iteration operator. It means that a preceding symbol or sequence of symbols between the brackets can repeat from zero to many times.

+ is Kleene plus or positive iteration operator. It means, that a preceding symbol or sequence of symbols between the brackets can repeat from one to many times.

? is Kleene question mark operator. It means that a preceding symbol or sequence of symbols between the brackets can be used zero or one times at particular place of the expression.

3.4.3 The Terminal and Non Terminal Symbols Shared by the Grammars

The parts of speech and punctuation marks of the English language are used in the both grammars. We utilize their full names or abbreviations (given in the brackets) as non terminals in the grammars, as practically they are changed either to the English words based on their parts of speech or straightly to punctuation marks. The full list of the parts of speeches and abbreviations used in the grammars is presented in the Appendix E.

3.4.4 The Rules of the Grammar of the Keywords

The Grammar of the Keywords is context-sensitive, as many other grammars used for natural language. It contains the following rules.

1. $Topic : \rightarrow$ ① $Keyword$: |
   ② $AbbrPhrase$ $((PreKW)^?$ $Keyword)^?$: |
   ③ $PrepRelPhrase$:
2. $Keyword \rightarrow (NounDD)^? \ (AdjDD \mid PastP)^? \ (NounDD)^+$
3. $PreKW \rightarrow (PDT)^? \ DT \mid PossPr$
4. $AdjDD \rightarrow Adjective$
5. NounDD
   (a) $NounDD \rightarrow$ ①$Noun \mid$ ② $Abbr$
   (b) $(CC \mid Sep \mid : \mid DetRel \mid PrepRel \ ) \ (PreKW)^? \ NounDD \rightarrow$
   $(CC \mid Sep \mid : \mid DetRel \mid PrepRel \ ) \ (PreKW)^? \ PP$
   (c) $NounDD \ (CC \mid Sep \mid : \mid DetRel \mid PrepRel) \rightarrow$
   $PP \ (PP \mid CC \mid Sep \mid : \mid DetRel \mid PrepRel)$
6. Abbreviation Phrase
   $AbbrPhrase \rightarrow Keyword \ (LP)Abbr(RP)$
7. Relations

(a) Prepositional relation
(*NounDD* | *Abbr* | *AbbrPhrase*) *PrepRel*
(*PreKW*)$^?$ (*Keyword* | *Abbr* | *AbbrPhrase*) $\rightarrow$
(*NounDD* | *Abbr* | *AbbrPhrase*) (*to* |*Prep*)
(*PreKW*)$^?$ (*Keyword* | *Abbr* | *AbbrPhrase*)

(b) Detalisation relation
DetRel $\rightarrow$ *such as* | *for example* | *for instance* | *e.g.*

(c) Verbal relation
(*NounDD* | *Abbr* | *AbbrPhrase*) (*to* | ,)$^?$ *VerbRel*
(*PreKW*)$^?$ (*Keyword* | *Abbr* | *AbbrPhrase*) $\rightarrow$
(*NounDD* | *Abbr* | *AbbrPhrase*)(((*to*)$^?$ *Verb*) |((,)$^?$ *PP*))
(*PreKW*)$^?$ (*Keyword* | *Abbr* | *AbbrPhrase*)

8. Relation phrases

(a) *PrepRelPhrase* $\rightarrow$ (*Keyword* | *Abbr* | *AbbrPhrase*) *PrepRel*
(*PreKW*)$^?$ (*Keyword* | *Abbr* | *AbbrPhrase*)

(b) *DetRelPhrase* $\rightarrow$ (*Keyword* | *Abbr* | *AbbrPhrase*)(,)$^?$ *DetRel*
(*PreKW*)$^?$ (*Keyword* | *Abbr* | *AbbrPhrase*)

(c) *VerbRelPhrase* $\rightarrow$ (*Keyword* | *Abbr* | *AbbrPhrase*)(*to*)$^?$ *VerbRel*
(*PreKW*)$^?$ (*Keyword* | *Abbr* | *AbbrPhrase*)

Explanation and Examples

1. The first statement introduces the rules used for recognition of the topics from the ”Keywords” section of a short module template. We assume, that a topic’s title is followed by a colon (’:’).

   The rule 1.1 introduces topics, whose titles contain a single keyword.

   Examples: Systems Analysis:

   The rule 1.2 shows, that a topic’s title may start directly with an abbreviation phrase, which may be followed by a construction of pre-keyword+keyword.

   Example: Database Management Systems (DBMS) issues:

   The three rules, included in bullet 1.3, state, that a prepositional relation phrases may act as topic as well.

   Example: Data structures in computer memory:

2. The second statement introduces a rule for a keyword's recognition. In accordance with it, a keyword is a structure, which may start with a NounDD, possibly followed by AdjDD, and definitely containing in the least one NounDD.

   Example: referential integrity; transaction management; caching large data; CASE tool; SQL.

3. The third statement introduces the construction, which may precede a keyword. It consists of either a possessive pronoun, or a determiner, possible preceded by a predeterminer.

   Example: class String and *all its* methods; *the* file server model.

4. The fourth group of rules introduces the methods for AdjDD recognition. The rule shows, that any adjective is marked as AdjDD.

   Example: *top-down* data modelling.

5. The fifth group of rules provides the means for NounDD recognition. According to the rule 5a any noun or abbreviation is treated as NounDD. In addition to this a present participle is considered to be used as NounDD, if it is preceded (5b) or followed (5c) by a coordinating conjunction, semicolon, a full-stop, a colon or by a detalisation or prepositional relation and is possibly preceded by a pre-keyword.

   Example: a) database, SQL; b) ,caching; c) data modelling,.

6. The sixth rule defines an abbreviation phrase as a keyword, followed by an abbreviation taken into the parentheses.

   Example: structured query language (SQL).

7. The seventh group of statements consists of the rules enabling recognition of relations of the different types. The rule 7a introduces a prepositional relation as a preposition or particle "to" used between a pair of keywords, abbreviations or abbreviation phrases. The rule 7b lists the phrases, which are marked as detalisation relations. The rule 7c, that a present participle or a verb, possibly preceded by the particle "to", is annotated as verbal relation, if it is placed between a pair of keywords, abbreviations or abbreviation phrases.

Example: a) presentation *of* data, use *of* SQL; b) RDBMS *such as* Oracle; c) database querying *using* Data Manipulation Language.

8. The statements of the eighth group annotate the prepositional, detalisation and verbal phrases. According to the rule 8a, a prepositional phrase contains two keywords, abbreviations or abbreviation phrases, separated by a prepositional relation. A detalisation phrase consists of the two keywords, abbreviations or abbreviation phrases with a detalisation relation between them, which may be preceded by comma (rule 8b). The rule 8c shows, that a verbal phrase contains a pair of keywords, abbreviations or abbreviation phrases, divided by a verbal relation, possibly following the particle "to".

   Example: a) presentation of data, use of SQL; b) RDBMS such as Oracle; c) database querying using Data Manipulation Language.

3.4.5 The Rules of the Grammar of the LOs

To improve of readability, the rules are divided into four major groups in accordance with their functional and structural roles in the learning outcomes. Thus to refer to a particular statement in the grammar, it is necessary to name its group and the number inside this group. In order to refer to a particular production inside the group's description, we use two numbers separated by a full-stop. The number before the full-stop refers to a particular statement inside the group of the rules. The number after the full-stop identifies the particular production inside the statement. When addressing the rule from other sections and chapters, we use the full pass, which includes group, the number of statement inside the group and the number of production inside the statement.

The groups and the sets of corresponding non-terminal symbols are listed below.

Group 1 The learning outcome sentence

The first group contains the axiom and other largest structural units of the GLO.

1. $NLLOSentence \rightarrow$ ① LO | ② LO (CC |, | Sep) $NLLOSentence$ | ③LO (LP)LO(RP)
2. $LO \rightarrow$ ① $MainLO$ | ② $MainLO((,)^{?}\ SClause)^{*}$

3. Main clause of the learning outcome.
   (a) *MainLO* → ①*AVC DDOC* ②*AVC WH DDOC AVCAdd*
   (b) (*LP*)(*SC*)$^?$*MainLO* → (*LP*)(*SC*)$^?$*AVC DDOC*
4. *WH* → *how* | *why* | *when* | *where* | *what*

Explanation and Examples

1. NLLOSentence stands for a natural language sentence, describing the learning outcomes (LO). This means that each NLLOSentence may contain in itself one (rule 1.1) or several learning outcomes (1.2). Each of the LOs is introduced by a separate clause. One clause is separated from another by a coordinate conjunction, comma or a Separator. One sentence may also contain two learning outcomes, where the second LO is taken into parentheses.

   Below are examples of sentences, each of which contains just one learning outcome (rule 1.1, examples a-d), two learning outcomes, connected by the coordinate conjunction "and" (rule 1.2, example e), two learning outcomes, where the second is in the brackets (rule 1.3, example f).

   Example:

   a Implement and test a small database application.
   b Critically evaluate and discuss key database technologies, management systems issues and database environments.
   c Create a range of standard SQL scripts to create and manipulate tables, adding and interrogating data.
   d Describe how analysis and design activities are organized within a leading systems development methodology (e.g. Unified software development process).
   e Map the design to a relational database management system such as Access and produce non-trivial queries to meet user requirements, using SQL.
   f Construct a set of analysis and design models using an appropriate range of object-modelling techniques that represent different perspectives of a particular systems development (e.g. Build UML models, such as use cases, class diagrams, activity diagrams, interaction diagrams, state charts and package diagrams).

2. LO — a learning outcome. According to the statement 2, each of the learning outcomes can contain either only the main clause (MainLO) (rule 2.1), or the main part and subordinating clause (-s), which specifies (-y) the main clause in some way (rule 2.2).

   Example:

   a-b The examples a and b consist of the main clauses only (rule 2.1).

   c-f The other examples contain the main parts and specifying clauses, which are emphasised with *italics*.

   c [MainLO] Create a range of standard SQL scripts [SClause 1]*to create and manipulate tables*, [SClause 2] *adding and interrogating data*.

   d [MainLO] Describe how analysis and design activities are organized [SClause 1] *within a leading systems development methodology* [SClause 2](*e.g. Unified software development process*).

   e [MainLO] Map the design to a relational database management system [SClause 1.1] *such as Access* and [MainLO] produce non-trivial queries [SClause 2.1] *to meet user requirements*, [SClause 2.2] *using SQL*.

   f [MainLO] Construct a set of analysis and design models [SClause 1.1] *using an appropriate range of object-modelling techniques* [SClause 1.2]*that represent different perspectives of a particular systems development* (e.g. [MainLO] Build UML models, [SClause 2.1] *such as use cases, class diagrams, activity diagrams, interaction diagrams, state charts and package diagrams*).

3. MainLO — the main part of a learning outcome. The MainLO indicates the action(-s), which a student is supposed to be able to perform over an object (or a group of objects), which belong to the data domain of the educational module. According to the statement 3, a MainLO can be structured in one of the following ways.

   The first type consists of the action verb construction (AVC) followed by a data domain object construction (DDOC) (rule 3a.1).

   Example:

   a [MainLO [AVC] *Implement and test* [DDOC] *a small database application*].

b [MainLO [AVC] *Critically evaluate and discuss* [DDOC] *key database technologies, management systems issues and database environments*].

c [MainLO [AVC] *Create a range of* [DDOC] *standard SQL scripts*] to create and manipulate tables, adding and interrogating data.

e [MainLO [AVC] *Map* [DDOC] *the design to a relational database management system*] such as Access and [MainLO [AVC] *produce* [DDOC] *non-trivial queries*] to meet user requirements, using SQL.

f [MainLO [AVC] *Construct* [DDOC] *a set of analysis and design models*] using an appropriate range of object-modelling techniques that represent different perspectives of a particular systems development (e.g. Build UML models, such as use cases, class diagrams, activity diagrams, interaction diagrams, state charts and package diagrams).

The second type of the structure (rule 3a.2) introduces the main clauses of the learning outcomes that contain the AVC, which indicates the action of a student, and the detalisation of the action in the passive voice (AVCAdd). They are divided by a WH and data domain object construction.

Example:

d [MainLO [AVC] *Describe* [WH] *how* [DDOC] *analysis and design activities* [AVCAdd] *are organized*] within a leading systems development methodology (e.g. Unified software development process).

The rule 3b additionally allows to recognise the MainLO, in case the learning outcome in the NLLOSentence appears in the brackets.

f Construct a set of analysis and design models using an appropriate range of object-modelling techniques that represent different perspectives of a particular systems development (e.g. [MainLO [AVC] *Build* [DDOC] *UML models*], such as use cases, class diagrams, activity diagrams, interaction diagrams, state charts and package diagrams).

4. *WH* is used to mark adverbs, starting with ”wh” and ”how”.

Group 2 Action

The second group of the rules describes the structural units, which are used to express the action, which a student should learn to perform over

the Object in the data domain of the educational module. It contains the following non-terminal symbols.

1. $AVC \rightarrow AV((, \mid CC \mid, CC)\ AVC)^*$
2. $AV \rightarrow$ ①$AdvVerb \mid$
   ② $AdvVerb\ (((PDT)^?\ DT) \mid PossPr)^?\ (Adj)^?\ NounCN\ (Prep)^?$
3. $AdvVerb \rightarrow$ ① $(MV)^?\ (Adv)^?\ ((Verb)^+ \mid OntAV) \mid$
   ② $(MV)^?\ ((Verb)^+ \mid OntAV)\ (Adv)^?$
4. $AVCAdd \rightarrow AVAdd((, \mid CC \mid, CC)\ (AV \mid AVAdd))^*$
5. $AVAdd \rightarrow (BE)\ PastP$
6. *OntAV* is an action verb, received from the ontology of verbs, which is described in detail in section 4.4.3.
7. *NounCN* → *range* | *need* | *relationship* | *link* | *understanding* | *experience*

   NounCN is a noun or an expression with a noun as a basic word, which should be followed by a noun only. This feature is observed in each of the GLOs rules, which contain NounCN.
8. *BE* stands for any form of the verb "to be" (e.g. is, are, were etc.).

Explanation and Examples

1. AVC — action verb construction. This is a sequence of expressions, separated by a comma or by a coordinating conjunction "and", which are used to describe the actions, which should be performed over a data domain object. The action verb constructions are emphasised with the help of *italics* in the examples below.

   Example:

   a [AVC] *Implement and test* a small database application.

   b [AVC] *Critically evaluate and discuss* key database technologies, management systems issues and database environments.

   c [AVC] *Create a range of* standard SQL scripts to [AVC] *create and manipulate* tables, adding and interrogating data.

   d [AVC] *Describe* how analysis and design activities are organized within a leading systems development methodology (e.g. Unified software development process).

e [AVC] *Map* the design to a relational database management system such as Access and [AVC] *produce* non-trivial queries to [AVC] *meet* user requirements, using SQL.

f [AVC] *Construct* a set of analysis and design models using an appropriate range of object-modelling techniques that [AVC] *represent* different perspectives of a particular systems development (e.g. [AVC] *Build* UML models, such as use cases, class diagrams, activity diagrams, interaction diagrams, state charts and package diagrams).

2. AV — action verb. This is a separate expression, which outlines a single action, which a student should be able to perform over some object on completion of the educational module or a course. In accordance with the statement 2, the action verb can be presented as a construction, built in one of the following ways.

   The rule 2.1 presents action verbs, which consist of a verb, which may be preceded or followed by an adverb.

   Example:

   a [AV] *Implement* and [AV] *test* a small database application.

   b [AV] *Critically evaluate* and [AV]*discuss* key database technologies, management systems issues and database environments.

   c Create a range of standard SQL scripts to [AV] *create* and [AV] *manipulate* tables, adding and interrogating data.

   d [AV] *Describe* how analysis and design activities are organized within a leading systems development methodology (e.g. Unified software development process).

   e [AV] *Map* the design to a relational database management system such as Access and [AV] *produce* non-trivial queries to [AV] *meet* user requirements, using SQL.

   f [AV] *Construct* a set of analysis and design models using an appropriate range of object-modelling techniques that [AV] *represent* different perspectives of a particular systems development (e.g. [AV] *Build* UML models, such as use cases, class diagrams, activity diagrams, interaction diagrams, state charts and package diagrams).

The rule 2.2 introduces more complicated action verbs. In this case the combination of a verb and an adverb is followed by a noun phrase. The noun phrase may start with a predeterminer+determiner or a possessive pronoun, possibly followed by an adjective. Afterwards a noun from the cognitive domain comes, which may be followed by a preposition. The example of such construction is given in the sentence c.

c [AV [Verb] *Create* [DT] *a* [NounCN] *range* [Prep] *of*]standard SQL scripts to [AV] *create* and [AV] *manipulate* tables, adding and interrogating data.

3. AdvVerb — construction, built from a verb and an adverb. In accordance with the statement 3, this construction may contain either a verb or a verb, which is preceded (rule 3.1) or followed (rule 3.2) by an adverb. It may also contain a modal verb (MV).

   Example:

   a [AdvVerb [Verb] *Implement*] and [AdvVerb [Verb] *test*] a small database application.

   b [AdvVerb [Adverb] *Critically* [Verb] *evaluate* and [AdvVerb [Verb] *discuss*] key database technologies, management systems issues and database environments.

   c Create a range of standard SQL scripts to [AdvVerb [Verb] *create*] and [AdvVerb [Verb] *manipulate*] tables, adding and interrogating data.

   d [AdvVerb [Verb] *Describe*] how analysis and design activities are organized within a leading systems development methodology (e.g. Unified software development process).

   e [AdvVerb [Verb] *Map*] the design to a relational database management system such as Access and [AdvVerb [Verb]*produce*] non-trivial queries to [AdvVerb [Verb] *meet*] user requirements, using SQL.

   f [AdvVerb [Verb] *Construct*] a set of analysis and design models using an appropriate range of object-modelling techniques that [AdvVerb [Verb] *represent*] different perspectives of a particular systems development (e.g. [AdvVerb [Verb] *Build*] UML models, such as use cases, class diagrams, activity diagrams, interaction diagrams, state charts and package diagrams).

4. The rule 4 introduces the sequences of the passive voice constructions.
   d Describe how analysis and design activities [AVCAdd *are organized*] within a leading systems development methodology (e.g. Unified software development process).
5. The rule 5 represents single passive voice constructions. The contain the verb "to be" in any form and a past participle.
   d Describe how analysis and design activities [AVCAdd [BE] *are* [PastP] *organized*] within a leading systems development methodology (e.g. Unified software development process).

Group 3 Object

The third group includes the units, which can be used to represent an object in the data domain of an educational module. They are as following.

1. $DDOC \rightarrow DDO((, \mid CC \mid CC)\ DDOC)^*$
2. $DDO \rightarrow Keyword_{LO}\ (Prep|to)\ Keyword_{LO}$
3. Keyword (in the terms of learning outcome)
   $Keyword_{LO} \rightarrow (((PDT)^?\ DT) \mid PossPr)^?$
   $(Char)^*\ (AdjDD \mid PastP)^?\ (NounDD\ |((LP)^? Abbr(RP^?)))^+$
4. $AdjDD \rightarrow Adjective \mid AdjOnt$
5. *AdjOnt* is an adjective, added to the data domain ontology at the stage of keywords recognition.
6. *NounDD* is a word, which is marked with the non terminal *Noun*, but neither belongs to *NounCN*, nor is included into *SC* or *OntAV*.
7. $Char \rightarrow$
   ① $Complexity\ Importance^?\ Size^?$ |
   ② $Complexity\ Size^?\ Importance^?$ |
   ③ $Importance\ Complexity^?\ Size^?$ |
   ④ $Importance\ Size^?\ Complexity^?$ |
   ⑤ $Size\ Complexity^?\ Importance^?$ |
   ⑥ $Size\ Importance^?\ Complexity^?$
8. $Complexity \rightarrow complicated \mid simple \mid standard \mid of\ middle\ complexity \mid$
   $of\ a\ given\ complexity \mid moderately\ complex \mid non-trivial$

9. *Importance* $\rightarrow$ *key* | *significant* | *insignificant*
10. *Size* $\rightarrow$ *small* | *large*

Explanation and Examples

1. DDOC — data domain object construction. This is a sequence of expressions, separated by a comma or by a coordinating conjunction "and", which are used to describe an object in the data domain of an educational module. Data Domain Object Construction (DDOC) defines the object or a set of objects, over which a student should be able to perform an action expressed by the Action Verb Construction previously defined in the Group 2. In the examples below the DDOCs are emphasised using *italics*. The details of how these structures are built can be found in the following points.

   Example:

   a Implement and test [DDOC] *a small database application*.

   b Critically evaluate and discuss [DDOC] *key database technologies, management systems issues and database environments*.

   c Create a range of [DDOC] *standard SQL scripts* to create and manipulate [DDOC] *tables*, adding and interrogating [DDOC] *data*.

   d Describe how [DDOC] *analysis and design activities* are organized within [DDOC] *a leading systems development methodology* (e.g. [DDOC] *Unified software development process*).

   e Map [DDOC] *the design to a relational database management system* such as [DDOC] *Access* and produce [DDOC] *non-trivial queries* to meet [DDOC] *user requirements*, using [DDOC] *SQL*.

   f Construct [DDOC] *a set of analysis and design models* using [DDOC] *an appropriate range of object-modelling techniques* that represent [DDOC] *different perspectives of a particular systems development* (e.g. Build [DDOC] *UML models*, such as [DDOC] *use cases, class diagrams, activity diagrams, interaction diagrams, state charts and package diagrams*).

2. DDO — data domain object. This is a sequence of words, which describes a single object in the data domain of an educational module.

According to the 2, the data domain object can be presented as a sequence of keywords (in the terms of learning outcomes) divided by prepositions or the particle "to".

Example:

a Implement and test [DDO] *a small database application*.

b Critically evaluate and discuss [DDO] *key database technologies*, [DDO] *management systems issues* and [DDO] *database environments*.

c Create a range of [DDO] *standard SQL scripts* to create and manipulate [DDO] *tables*, adding and interrogating [DDO] *data*.

d Describe how [DDO] *analysis* and [DDO] *design activities* are organized within [DDO] *a leading systems development methodology* (e.g. [DDO] *Unified software development process*).

e Map [DDO] *the design to a relational database management system* such as [DDO] *Access* and produce [DDO] *non-trivial queries* to meet [DDO] *user requirements*, using [DDO] *SQL*.

f Construct [DDO] *a set of analysis* and [DDO] *design models* using [DDO] *an appropriate range of object-modelling techniques* that represent [DDO] *different perspectives of a particular systems development* (e.g. Build [DDO] *UML models*, such as [DDO] *use cases*, [DDO] *class diagrams*, [DDO] *activity diagrams*, [DDO] *interaction diagrams*, [DDO] *state charts* and [DDO] *package diagrams*).

3. The third rule defines a keyword in the terms of learning outcomes. According to it, it contains a sequence of nouns with optional abbreviation given in the brackets. The sequence maybe preceded either by predeterminer+determiner or a possessive pronoun, which may be followed by characteristic and an AdjDD or single past participle.

Example:

a Implement and test [$Keyword_{LO}$] *a small database application*.

b Critically evaluate and discuss [$Keyword_{LO}$] *key database technologies*, [$Keyword_{LO}$] *management systems issues* and [$Keyword_{LO}$] *database environments*.

c Create a range of [$Keyword_{LO}$] *standard SQL scripts* to create and manipulate [$Keyword_{LO}$] *tables*, adding and interrogating [$Keyword_{LO}$] *data*.

d Describe how[$Keyword_{LO}$] *analysis* and [$Keyword_{LO}$] *design activities* are organized within [$Keyword_{LO}$] *a leading systems development methodology* (e.g. [$Keyword_{LO}$] *Unified software development process*).

e Map [$Keyword_{LO}$] *the design* to [$Keyword_{LO}$] *a relational database management system* such as [$Keyword_{LO}$] *Access* and produce [$Keyword_{LO}$] *non-trivial queries* to meet [$Keyword_{LO}$] *user requirements*, using [$Keyword_{LO}$] *SQL*.

f Construct [$Keyword_{LO}$] *a set* of [$Keyword_{LO}$] *analysis* and [$Keyword_{LO}$] *design models* using [$Keyword_{LO}$] *an appropriate range* of [$Keyword_{LO}$] *object-modelling techniques* that represent [$Keyword_{LO}$] *different perspectives* of [$Keyword_{LO}$] *a particular systems development* (e.g. Build [$Keyword_{LO}$] *UML models*, such as [$Keyword_{LO}$] *use cases*, [$Keyword_{LO}$] *class diagrams*, [$Keyword_{LO}$] *activity diagrams*, [$Keyword_{LO}$] *interaction diagrams*, [$Keyword_{LO}$] *state charts* and [$Keyword_{LO}$] *package diagrams*).

4. Char — is an adjective or a sequence of adjectives, which characterise the data domain object in one of the following ways:

5. Complexity.

6. Importance.

7. Size.

Example:

a Implement and test a [Char] *small* database application.
Here the adjective "small" characterises the Size.

b Critically evaluate and discuss [Char] *key* database technologies, management systems issues and database environments.
In this example the adjective "key" means, that the student is to know only the most important database issues and is recognised as Importance.

c Create a range of [Char] *standard* SQL scripts to create and manipulate tables, adding and interrogating data.

d Describe how analysis and design activities are organized within a [Char] *leading* systems development methodology (e.g. Unified software development process).
The adjective "leading" reflects Importance.

f Map the design to a relational database management system such as Access and produce [Char] *non-trivial* queries to meet user requirements, using SQL.
In this example "non-trivial" again represents the Complexity of the queries to be produced.

Group 4 Specifying Clause

This group contains the constructions, which can be used in the specifying clauses.

1. *SClause* → ① (*SC*|*WH*) *AVC DDOC* | ② *SC DDOC* |
③ (*LP*)*SC DDOC*(*RP*) ④ (*WH*)$^{?}$ *PP*((, | *CC*) *PP*)* *DDOC*
2. *SC* → *such as* | *based on* | *with respect to* |
given | *e.g.* | *within* | *that* | *for* | *given in*

Explanation and Examples

1. SClause — a specifying clause. This is a construction, which clarifies and expands the sense of the main sentence of the learning outcome. The statement proposes four rules for the clauses recognition.
The first rule introduces specifying clauses, which start either with subordinating conjunction or a wh-adverb and contain the action verb and data domain object constructions.
Example:
c Create a range of standard SQL scripts [SClause [SC] *to* [AVC] *create and manipulate* [DDOC] *tables*], adding and interrogating data.
e Map the design to a relational database management system such as Access and produce non-trivial queries [SClause [SC] *to* [AVC] *meet* [DDOC] *user requirements*], using SQL.
f Construct a set of analysis and design models using an appropriate range of object-modelling techniques [SClause [SC] *that* [AVC] *represent* [DDOC] *different perspectives of a particular systems development*] (e.g. Build UML models, such as use cases, class diagrams, activity diagrams, interaction diagrams, state charts and package diagrams).

The second rule shows, that a specifying clause may consist of a subordinating conjunction/ wh-adverb and a data domain object construction.

d Describe how analysis and design activities are organized [SClause [SC] *within* [DDOC] *a leading systems development methodology*] (e.g. Unified software development process).

e Map the design to a relational database management system [SClause [SC] *such as* [DDOC] *Access*] and produce non-trivial queries to meet user requirements, using SQL.

f Construct a set of analysis and design models using an appropriate range of object-modelling techniques that represent different perspectives of a particular systems development (e.g. Build UML models, [SClause [SC] *such as* [DDOC] *use cases, class diagrams, activity diagrams, interaction diagrams, state charts and package diagrams*]).

The third rule says, that a specifying clause, consisting of a subordinating conjunction/ wh-adverb and a data domain object construction, may be taken into parentheses.

d Describe how analysis and design activities are organized within a leading systems development methodology [SClause [LP]([SC] *e.g.* [DDOC] *Unified software development process*[RP])].

The fourth rule introduces specifying clauses, which start with a sequence of present participles followed by a data domain object construction. The wh-adverb may appear before the participle.

c Create a range of standard SQL scripts to create and manipulate tables, [SClause [PP] *adding* [CC] *and* [PP] *interrogating* [DDOC] *data*].

e Map the design to a relational database management system such as Access and produce non-trivial queries to meet user requirements, [SClause [PP] *using* [DDOC] *SQL*].

f Construct a set of analysis and design models [SClause [PP] *using* [DDOC] *an appropriate range of object-modelling techniques*] that represent different perspectives of a particular systems development (e.g. Build UML models, such as use cases, class diagrams, activity diagrams, interaction diagrams, state charts and package diagrams).

3.5 The Data Domain Ontology

We assume that the users of this software are neither experts in ontology building, nor in the data domain fields of the educational modules and courses. This assumption is reasonable, as we suppose to use our approach at the earlier stage of educational modules' and courses' comparison in order to reject those, which are definitely not comparable and to distinguish the alike pairs, providing the level of confidence in each alignment.

We suppose that the documents are the only sources of information for ontology creation. The analysis of the module templates showed, that they contain the information, which describe the data domain of a module, in the sections called "Keywords" (British templates) and "Structure of the Module" (Russian templates), which look and serve almost the same purposes despite the different titles (see chapter 2). These parts of the documents contain the lists of topics, each of which includes the sets of keywords and collocations. This information enables extraction of the key concepts of the data domain, but provides poor information on the relations between them. For this reason we decided to create a unified top-level data domain ontology, which can store information on any data domain based on the data extracted from the documents with particular above described structure.

3.5.1 The Formal Model of the Data Domain

Data domain of an educational module is described by the following 5-uple.

$$DD = (DDTitle, Topic, Keyword, \alpha, \alpha^{-1}) \tag{3.10}$$

The Definition 3.10 has the following constituents.

1. *DDTitle* contains the title of the data domain.
2. $Topic = \{Topic_1, \ldots, Topic_i, \ldots, Topic_m\}$ is the set of data domain topics, which are studied in the educational module.

3. $Keyword = \{Keyword_{Topic_1}, \ldots, Keyword_{Topic_i}, \ldots, Keyword_{Topic_m}\}$ is the set of subsets of the keywords, divided in accordance with their pertinence to certain topics. Here each subset contains particular keywords. $Keyword_{Topic_j} = \{Keyword_1, \ldots, Keyword_j, \ldots, Keyword_n\}$

4. $Keyword_j = ([AdjDD], NounDD_{j_1}, \ldots, NounDD_{j_i})$

5. $\alpha : Topic_i \rightarrow Keyword_{Topic_i}$ is the function, which maps each of the topics to a particular set of keywords.

6. $\alpha^{-1} : Keyword_j \rightarrow Topic_i$ is the inverse function to α. It builds a link between a keyword an d a topic, which contains this concept.

The top-level ontology of the data domain of an educational module contains the following classes and properties.

3.5.2 Creation of the Ontology of the Data Domain based on the Formal Model

Classes

The ontology contains the following classes.

1. **Keyword** contains the keywords from the topics and collocations.

2. **Topic** contains the titles of the topics, studied in the educational module. The **Topic** is a subclass of the **Keyword**, as any topic is also a keyword and thus has the same properties as it.

3. **NounDD** contains the one-word keywords, which are nouns and the single nouns from the collocations. It is a subclass of the class **Keyword**.

4. **AdjectiveDD** contains the one-word keywords, which are adjectives and the single adjectives from the collocations. It is a subclass of the class **Keyword**.

The ontology relations are described in detail in Appendix C.

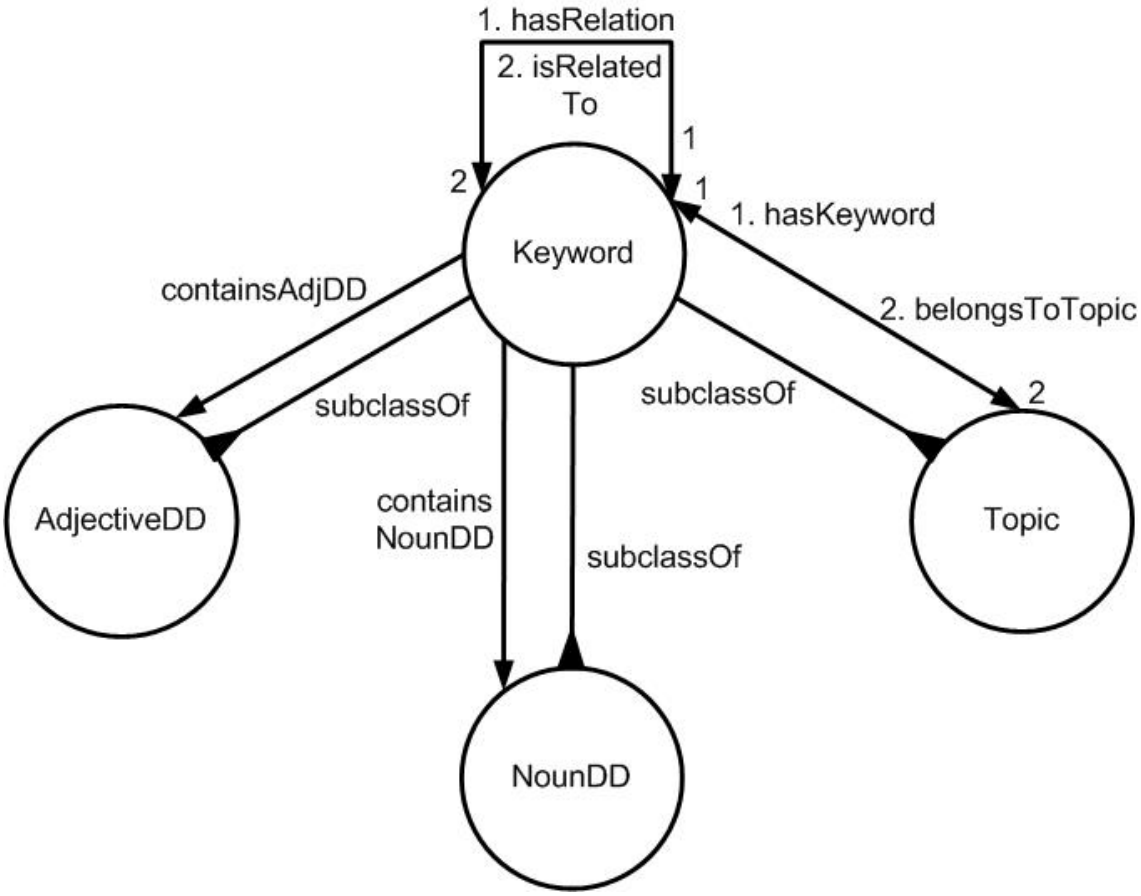

Figure 3.3: The Ontology of the Data Domain

3.6 Ontology of a Learning Outcome

3.6.1 The Formal Model of a Learning Outcome

The formal model of a learning outcome represents only the part of the grammar of a learning outcome. The reason for this lies in their different functional application. The formal model is used as a basis for the ontology, which in turn serves the purpose of comparing of the learning outcomes. The grammar is used to recognize the ontology's individuals from the natural language text, which contains much more information (practically, extra words), than is necessary for the task of comparison.

In the formal model we treat a learning outcome as the combination of an action verb (group 2 Action of the GLO) and a data domain object (group 3 Object of the

GLO) constructions and possibly specifying clauses (group 4 Specifying Clause of the GLO). All these concepts were described in detail in the previous section.

$$LO = (AVC, DDOC[, SClauses]) \tag{3.11}$$

The action verb construction is treated as a set of verbs, which carry the sense of the action. This means, that we do not need to include modal verbs and other words, which may be present in an action verb according to the GLO.

$$AVC = \{Verb_1, \ldots, Verb_i, \ldots, Verb_m\} \tag{3.12}$$

The data domain object construction is presented as the set of data domain objects.

$$DDOC = (DDO_1, \ldots, DDO_j, \ldots, DDO_n) \tag{3.13}$$

Like the notion of the action verb was reduced to the set of verbs, not all the constituents, defined in the GLO, are present in the formal model of the data domain object, which is used for building the ontology.

$$DDO_j = (NLPhrase, [AdjDD,]NounDDC[, Characteristics]) \tag{3.14}$$

The data domain object is defined in the Equation 3.14 as the following quadruple. Square brackets are used to mark up optional elements.

1. *NLPhrase* is the whole natural language phrase, defining the data domain object.
2. *AdjDD* is the data domain adjective, which may be present in the object.
3. *NounDDC* is the sequence of data domain nouns.
4. *Characteristics* is the set of words, specifying the data domain object in certain directions. It has the same notion as defined in p.7 of the group 3 Object of the GLO.

5. *SClauses* is the set of specifying clauses of a learning outcome, where each of them has the following structure.

$$SClause = ([SC][AVC,]DDOC) \tag{3.15}$$

Here *SC* stands for a subordinating conjunction, *AVC* and *DDOC* are action verb and data domain object constructions correspondingly.

3.6.2 Creation of the Ontology of a Learning Outcome

The ontology of a learning outcome is built upon the formal model and is presented in the Figure 3.4.

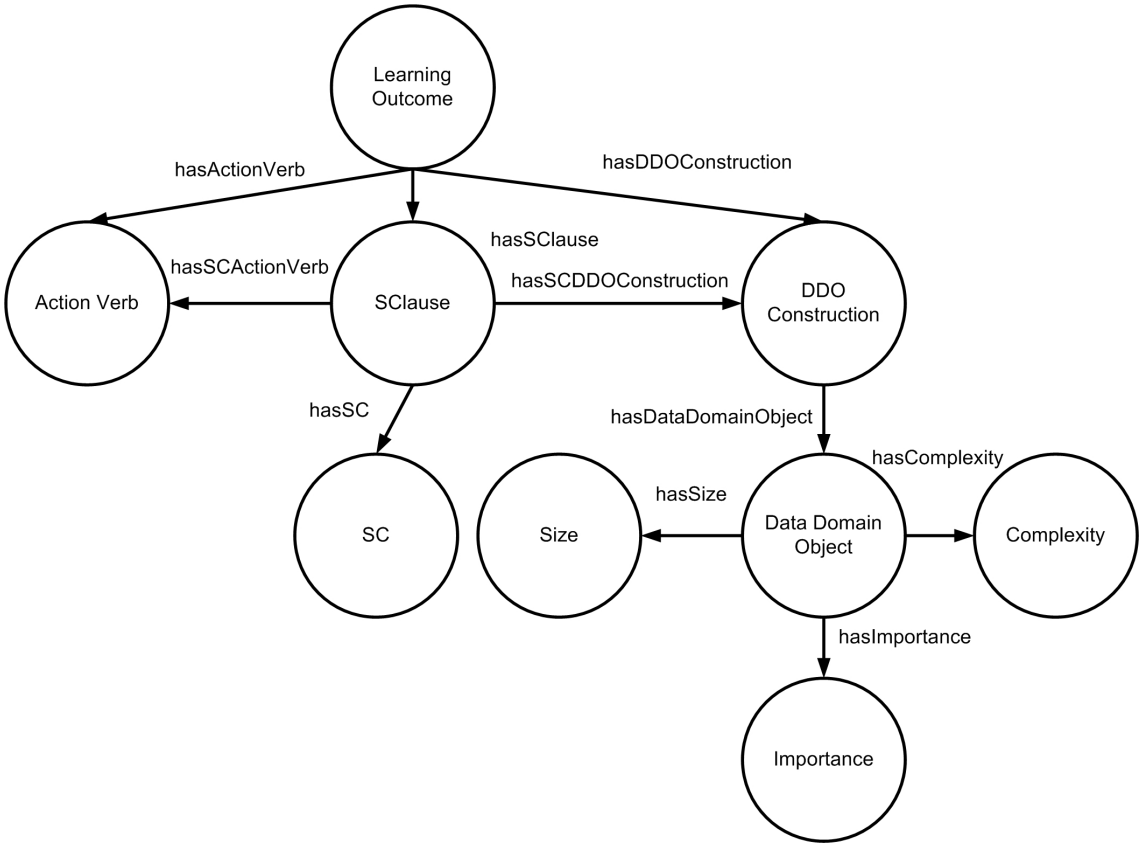

Figure 3.4: The Ontology of a Learning Outcome

Classes

The ontology contains the following classes.

1. **Learning Outcome** contains identifiers of the learning outcomes of an educational module.
2. **Action Verb** contains the action verbs of a learning outcome.
3. **DDOConstruction** contains identifiers of the data domain object constructions, which unite the data domain objects.
4. **DataDomainObject** contains the identifiers of the data domain objects, each of which is linked to a keyword from the ontology of the data domain of the educational module.
5. **Complexity**, **Importance**, **Size** contain as individuals the words, which characterize the data domain objects in the different aspects.
6. **SClause** contains identifier of the specifying clause for the learning outcome.
7. **SC** contains the subordinating constructions, which a specifying clause may start with.

The relations of the ontology of a learning outcome are described in Appendix D.

3.7 Combination of the Ontologies

The Figure 3.5 shows that the class **Learning Outcome** unites the basic course and module ontology and ontology of a learning outcome. At the same time *hasDDO (DataDomainObject, Keyword)* is the property linking the ontology of a learning outcome to the data domain ontology. It enables the choice of the data domain objects from the keywords, describing the field of study of the educational module. This property has the following restrictions.

1. *hasDDO.Keyword* means, that the concepts of the data domain object can be chosen only from the class **Keyword** of the data domain ontology.
2. $\geq$ 1. *hasDDO* means, that any data domain object must have in the least one keyword.

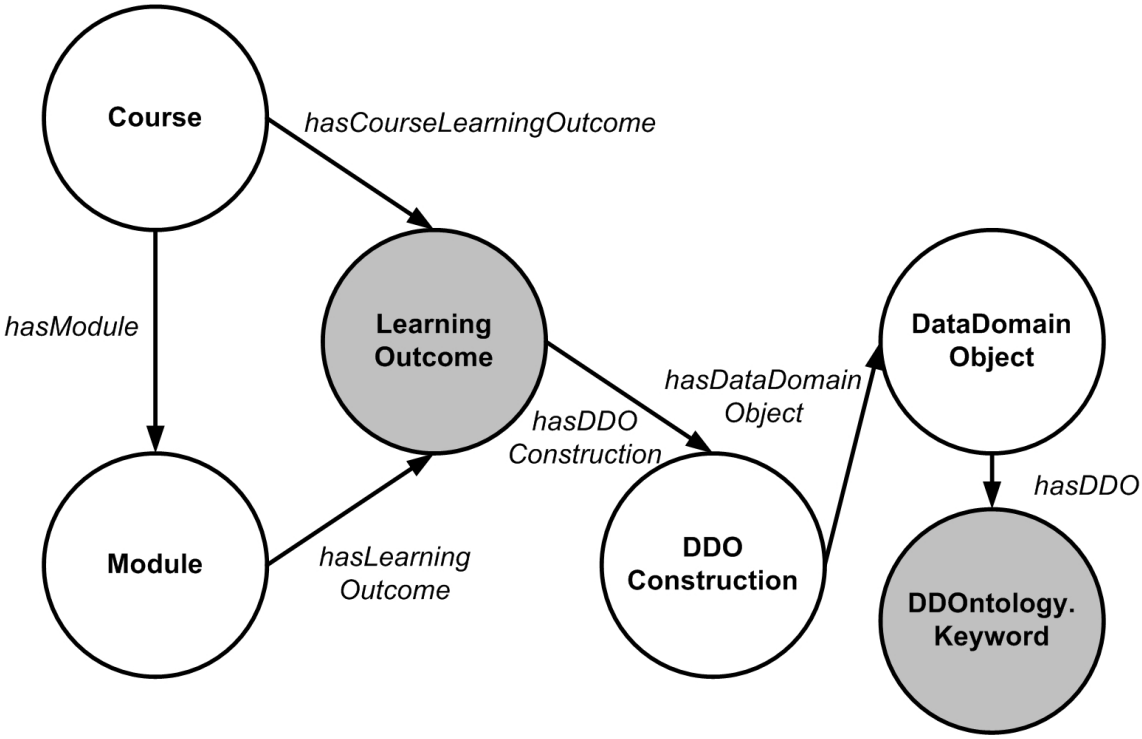

Figure 3.5: Combination of the Basic Course and Module, Learning Outcome and Data Domain Ontologies

3.8 Summary

This chapter was devoted to creation of the ontology for educational courses' and modules' representation. We outlined the ontology building and populating methodology. We proposed to model the general information about educational courses, modules and their interdependencies, the learning outcomes and the fields of study separately and add the links between these subdomains afterwards.

Firstly, we built the mathematical model of an educational course, the modules, their general properties and relations and turned it into ontology.

Secondly, we distinguished a learning outcome as an entity, which is to be modelled in a very detailed way. We also decided to design a separate unified ontology to describe the data domain of an educational module based on its keywords. Another concern was to provide the means to populate the learning outcomes' and field of study (sub)ontologies from the natural language texts.

The process was organised in the following way. At first we carefully studied the "Learning Outcomes" and "Keywords" sections from the module templates

from De Montfort University and learnt the main patterns. Based on them the grammars of the keywords and learning outcomes were designed. The grammars allow recognition and markup of each symbol in the sentences. However, not all the tagged information is necessary for comparison of the keywords and learning outcomes. For this reason we built the mathematical models of a keyword and a learning outcome to store only meaningful data. Afterwards the ontologies were created based on these models. Finally, we illustrated, how all the three parts (course and module, learning outcome and data domain subontologies) link together to reflect the whole picture of an educational course.

Chapter 4

Alignment Algorithm for the Ontologies of an Educational Course and Modules

4.1 Introduction

This chapter reveals the essence of this research work. It describes the methodology, which we use to characterize the similarity between educational courses through their educational modules.

We stated in Chapter 2 that programme specifications provide information at a high level of abstraction. To compare the contents of the educational courses we should analyze the modules, which they consist of. For this reason we treat the courses as the sets of modules. We aim to provide the algorithm for pairwise comparison of the educational modules and courses in this chapter.

Firstly, we present an overview of the approach and the methodology. Further on, we proceed with the algorithm for comparison of the educational modules and courses. We describe in detail each of its three major steps: comparison of the keywords, comparison of the learning outcomes and aggregation of the results. The novel similarity measure CAVe, which we designed especially for comparison of the learning outcomes' action verbs, is also depicted in this chapter.

4.2 The Alignment Methodology

In accordance with the formal model, we handle a programme specification as a set of educational courses, which can be built based on it by varying the choice of elective and optional educational modules. As it was concluded in the Chapter 2, the comparison of educational courses is based on collation of the sets of the keywords and the learning outcomes of their modules.

An educational course is treated as a fixed set of educational modules and thus it inherits all their keywords and learning outcomes. For this reason the task of the educational courses' comparison is analogous to comparison of the educational modules. The main difference between collation of the courses and the modules is the size of ontology, as a course contains all the keywords and learning outcomes of its components. Therefore we will describe in detail the alignment methodology for a pair of educational modules, considering that its main stages are the same for comparison of a pair of courses.

The alignment methodology for the ontologies consists of the following significant stages, presented in the Figure 4.1 and described below. Two educational modules (or courses) are comparable if both of them belong to the same educational level and cost the same number of credits. In case of comparison of educational courses we do not check this condition for each pair of the educational modules inside them. We assume that if they already belong to the course, then they are suitable for the certain level of education. We do not check the credits for equality, because our work is aimed at deeper analysis: we wish to find all the alike keywords and learning outcomes, independently on how they are spread among the modules.

1. The first step of the methodology is alignment of the data domain sub-ontologies, containing the topics and keywords of the pair of educational modules ($sim_{Kw_{aggr}}(M_1, M_2)$) or all the modules of a pair of educational courses ($sim_{Kw_{aggr}}(M_{C1}, M_{C2})$).

2. The second step is the alignment of the sets of the learning outcomes of the pair of modules ($sim_{LO_{aggr}}(M_1, M_2)$) or all the modules of a pair of educational courses ($sim_{LO_{aggr}}(M_{C1}, M_{C2})$).

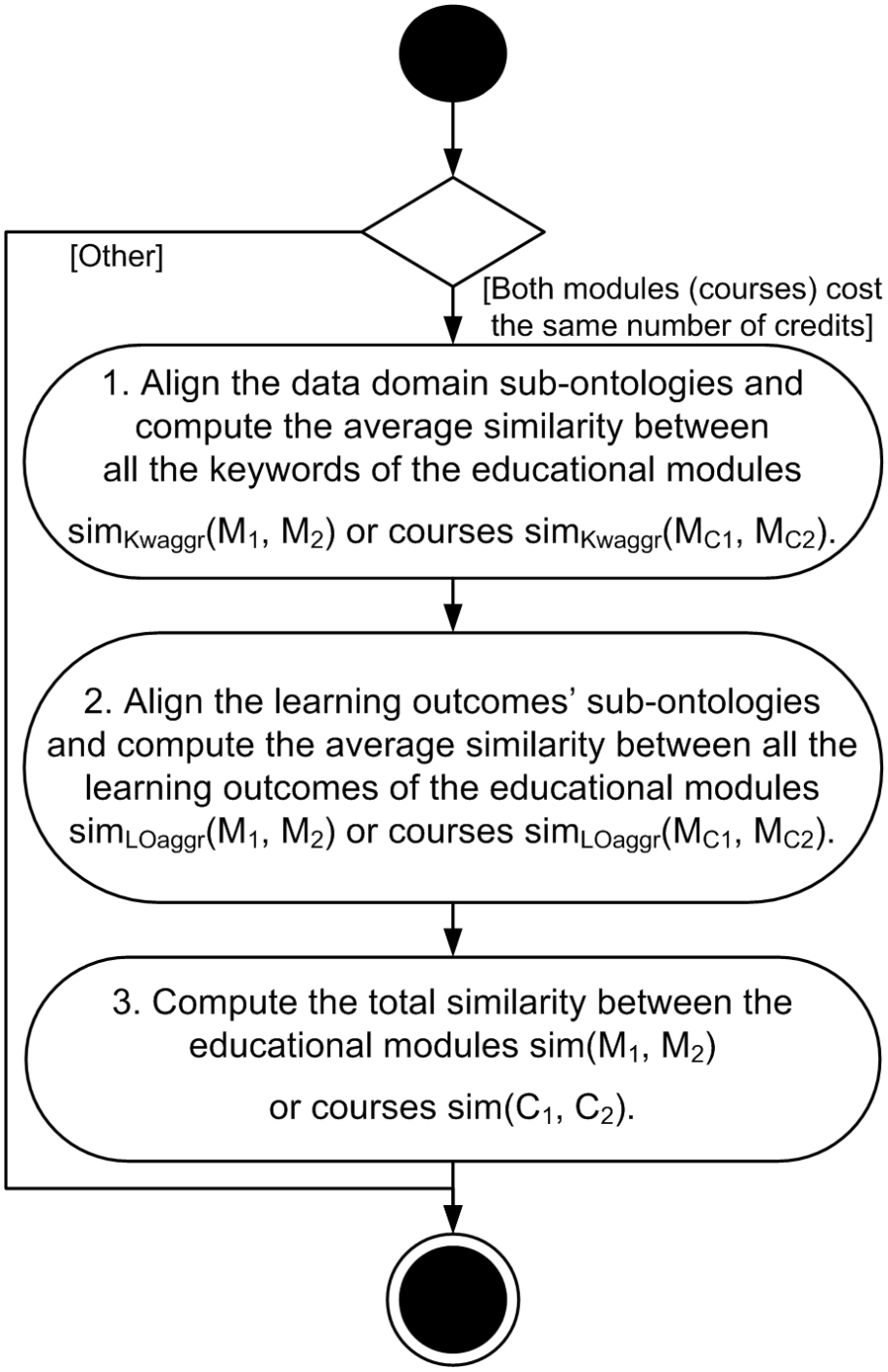

Figure 4.1: The Algorithm for Comparison of the Educational Modules (Courses) based on Ontology Alignment of Keywords and Learning Outcomes

3. The third step is devoted to aggregation of the results of the previous stages. The total similarity is calculated as an average between the similarity for the keywords and the learning outcomes.

$$sim(M_1, M_2) = \frac{sim_{Kw_{aggr}}(M_1, M_2) + sim_{LO_{aggr}}(M_1, M_2)}{2}. \quad (4.1)$$

Analogically, the similarity for educational courses equals the average between the similarities for the keywords and learning outcomes of their modules.

$$sim(C_1, C_2) = \frac{sim_{Kw_{aggr}}(M_{C1}, M_{C2}) + sim_{LO_{aggr}}(M_{C1}, M_{C2})}{2}. \quad (4.2)$$

We describe each of the major stages in the very detail in the following sections.

The next sections introduce some common approaches, used at several stages of the algorithm.

4.2.1 Choosing the Best Matches

The problem of choosing the best matching pairs of some entities from the matrices arises at several stages of the alignment algorithm. In order to do this, we designed two types of strategies.

Greedy Iterative Strategy

In some cases we need to select the pairs of words or word combinations with the maximum similarity values. Having studied the strategies, proposed in [39, pp. 72-73], we decided to introduce a selection method and called it the greedy iterative strategy. It was derived from the widely used greedy strategy. The essence of this method is to find the best matching pairs of words, on the one hand, having as few cases of comparing several entities from one side to the same entity on another as possible, and on the other hand, considering the different numbers of constituents in the compared instances.

The algorithm of the greedy iterative strategy is presented in the Figure 4.2 and described below.

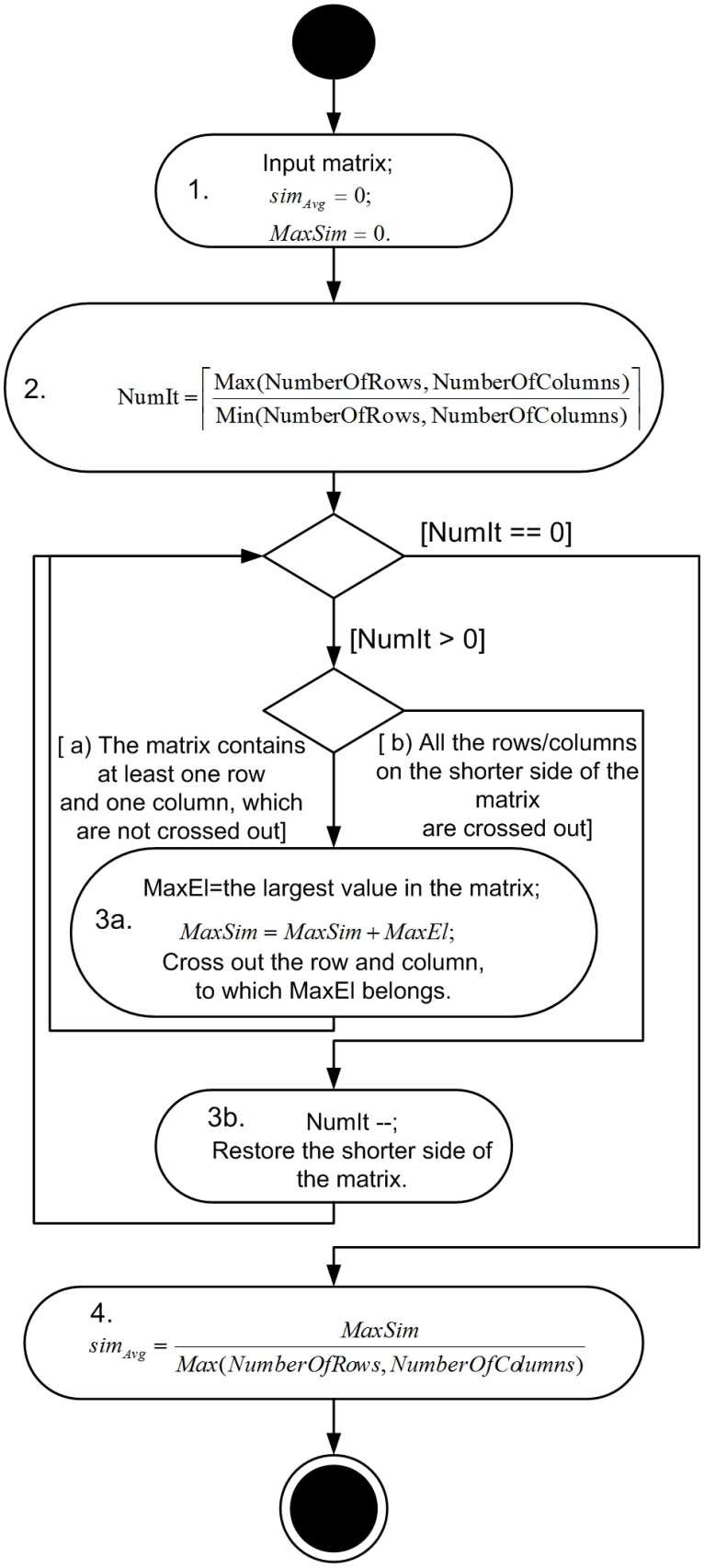

Figure 4.2: The Greedy Iterative Strategy

1. Set the resulting similarity value (sim_{Avg}) and the sum of the similarity values MaxSim to zero.

2. Calculate the number of iterations (*NumIt*) by dividing the longer side of the matrix by the shorter one. In case of fractional number, round it to the nearest larger integer value.

$$NumIt = \lceil (\frac{Max(NumberOfRows, NumberOfColumns)}{Min(NumberOfRows, NumberOfColumns)}) \rceil \tag{4.3}$$

3. Repeat the following steps *NumIt* times.

   (a) While the matrix contains at least one uncrossed pair of rows and columns, find the largest number in the matrix and add it to the sum of the similarity values MaxSim. Remove the row and the column, which are the coordinates of the maximum value.

   (b) If the matrix is not square (number of iterations is more than one, which means that the matrix contains more rows than the columns or vice versae), restore the shorter side and repeat the previous action for the remaining elements of the longer side. Decrement *NumIt*.

4. Receive the resulting value sim_{Avg} by dividing the sum of the similarity values by the size of the longer side of the matrix (the number of rows or columns).

$$sim_{Avg} = \frac{MaxSim}{Max(NumberOfRows, NumberOfColumns)} \tag{4.4}$$

   Further on in particular pieces of the algorithm we assume $MaxSim = \sum Max(sim)$, where *sim* stands for particular type of similarity metrics for a pair of entities, while *Max* means that the maximum similarity value is excerpted from the matrix at each iteration.

Maximum Average strategy

We designed this strategy in order to choose the best pairs from the matrices, containing similarities for complex entities, such as sets of keywords (when comparing two "Keywords" sections of the module templates) and sets of learning

outcomes (when comparing two "Learning Outcomes" sections of the module templates). The intuition behind this method is to find the most similar sentences or word sequences, not considering whether one or several entities from the one side are compared to the same entity on another side. The algorithm takes a matrix with m rows and n columns as an input and performs the following steps, presented in the Figure 4.3 and described below.

1. Set the threshold value τ. If the value in the cell is less than τ, it is reduced to zero.
2. For each side of the matrix (rows and columns consequently):
   (a) Initialize variable $\sum sim_{row}$ ($\sum sim_{column}$) with the value zero.
   (b) In each row (column) find the largest value. Add it to the value $\sum sim_{row}$ ($\sum sim_{column}$).
   (c) Divide the $\sum sim_{row}$ ($\sum sim_{column}$) by the number of rows (columns) and receive the values Sim_{RowAvg} ($Sim_{ColumnAvg}$) respectively.
3. Receive the final similarity value.

$$sim_{Avg} = \frac{Sim_{RowAvg} + Sim_{ColumnAvg}}{2}$$

4.2.2 The Similarity Measure for Comparison of the Words

In this subsection we describe the measure, which we utilize to compare two words, which can be either the whole keywords or the constituents of the key phrases, containing several words. It is also utilized as one of the similarity measures for the verbs of the learning outcomes, which is discussed in one of the following sections.

In order to compare two words, firstly, WordNet is utilized to find out if they are synonyms. In such a case the similarity between them is set to 1 and no further calculations follow. If the words are not synonyms, then the next similarity measures are used for comparison.

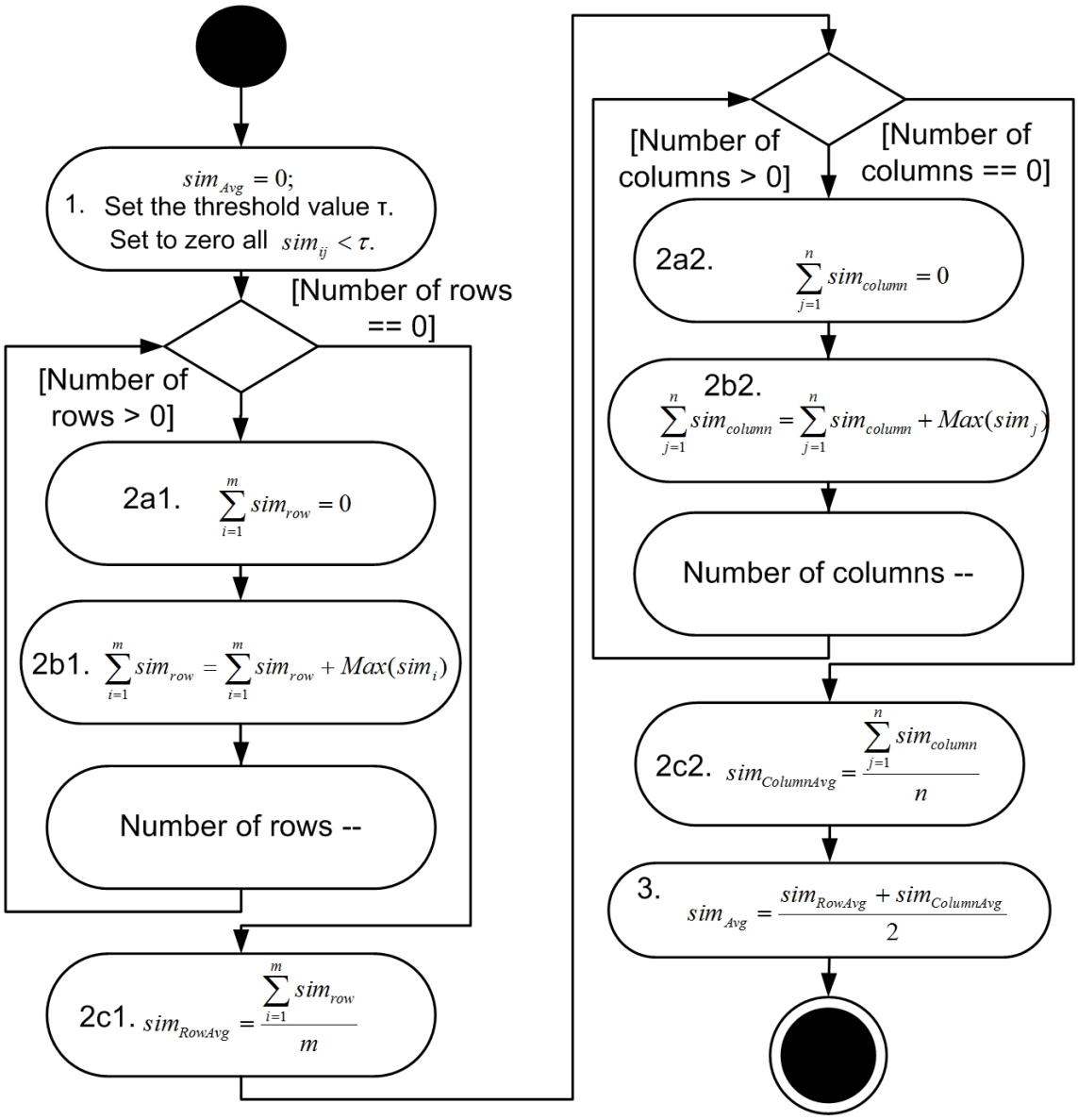

Figure 4.3: The Maximum Average Strategy

1. **Edit Distances-based Measures**. In our work we use the similarity measure based on Levenshtein edit distance and Soundex (with Jaro-Winkler). In accordance with triangular conorm, the maximum value of them is chosen for further calculations.

2. **Structure and Semantic Measure**. We chose Wu and Palmer measure, which calculates similarity between the words based on the distance between each of them and their closest common subsumer in WordNet hierarchy.

3. **Distributional Similarity/Relatedness Measure**. We utilized the distributional measure DISCO2. It calculates the similarity relatedness between the words based on their sets of distributionally similar words in the text corpuses.

We combined the measures, which characterise similarity between a pair of words from different points of view. The edit-distance-based measures ensure that we do not miss similar words, if they are misspelled. They also increase similarity value between the words containing common substrings (e.g. database and data). Wu and Palmer measure is used to evaluate semantic similarity between the two words based on hyponymy/hypernymy relations in the lexical database WordNet. DISCO2 measure shows, if the compared words and their collocations are used in the same text corpuses. To sum up, we consider if the compared words share common sequences of letters, are likely to have the same meaning and can be met together in the same texts. We assume that all these characteristics are equally important. For this reason we assign equal weights to each of the measures in the resulting aggregated similarity.

The aggregated similarity measure between two words is calculated in accordance with the following Formula.

$$sim_{wStrSem}(w_1, w_2) = \begin{cases} 1\text{,iff the words are synonyms} \\ \frac{sim_{str}(w_1,w_2)+sim_{WP}(w_1,w_2)+sim_{DISCO2}(w_1,w_2)}{3} \text{ ,in other case.} \end{cases} \tag{4.5}$$

4.3 The Alignment of the Keywords

The strategy for comparison of the two keywords is based on the assumption that they can be distributed among the topics in accordance with different criteria. For this reason for each of the keywords we compute similarities of two types. The first processes them as bags of words and counts their relations with other keywords in corresponding data domain ontologies. The second measure analyses the topics, to which the keywords belong. The algorithm for comparison of the keywords contains the following two steps: firstly, each keyword from one module is compared against every keyword of the second, secondly, the results are aggregated. The algorithm is presented in the Figure 4.4.

As can be seen from the Figure 4.4, similarity between the keywords is calculated in three steps. The first of them (step 2 in the Figure 4.4) is described in the Section 4.3.1. The second (step 3 in the Figure 4.4) is depicted in the Section 4.3.2. The total similarity for a pair of keywords (step 4 in the Figure 4.4) is presented in the Section 4.3.4. Afterwards we provide the Section 4.3.4, explaining how the results are aggregated (1, 5-6 in the Picture 4.4).

4.3.1 Comparison of the Pairs of Keywords

The algorithm for comparison of a pair of keywords, none of which are the topics, is presented in the Figure 4.5.

The first step of the algorithm is the following. In accordance with the ontological model, we address a keyword as a set of words, containing one or more nouns (individuals of the class **NounDD**) and possibly one adjective (individuals of the class **AdjectiveDD**) from the data domain of the educational module. The adjectives and nouns of the keywords are compared separately. If it is explicitly defined in the ontology that the keywords contain several nouns, then they are compared pairwise.

In the Table 4.1 the first row denotes the data domain adjective of the keyword from the first module template, the others stand for the data domain nouns of the keyword. In the same way the columns represent the adjective and nouns of the second keyword. The cells of Table 4.1 contain the values of the structural and

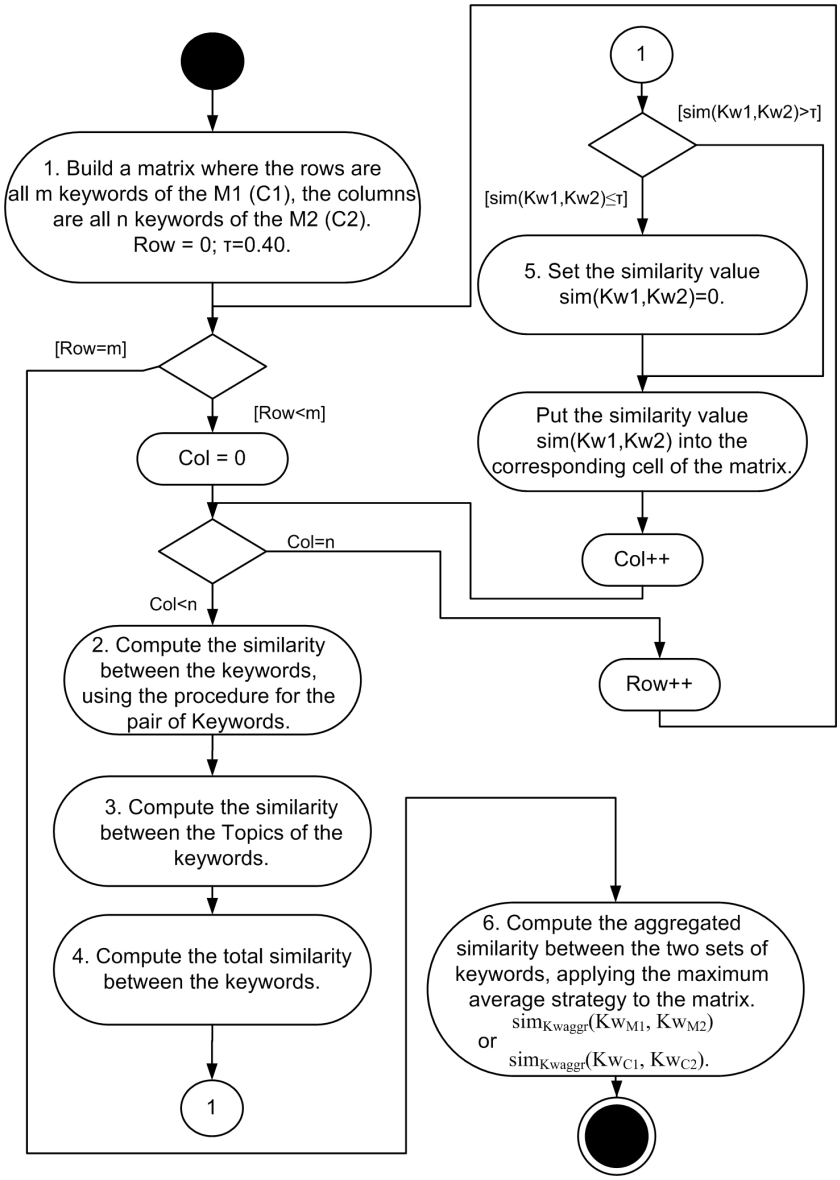

Figure 4.4: The Alignment of the Keywords of Modules (Courses)

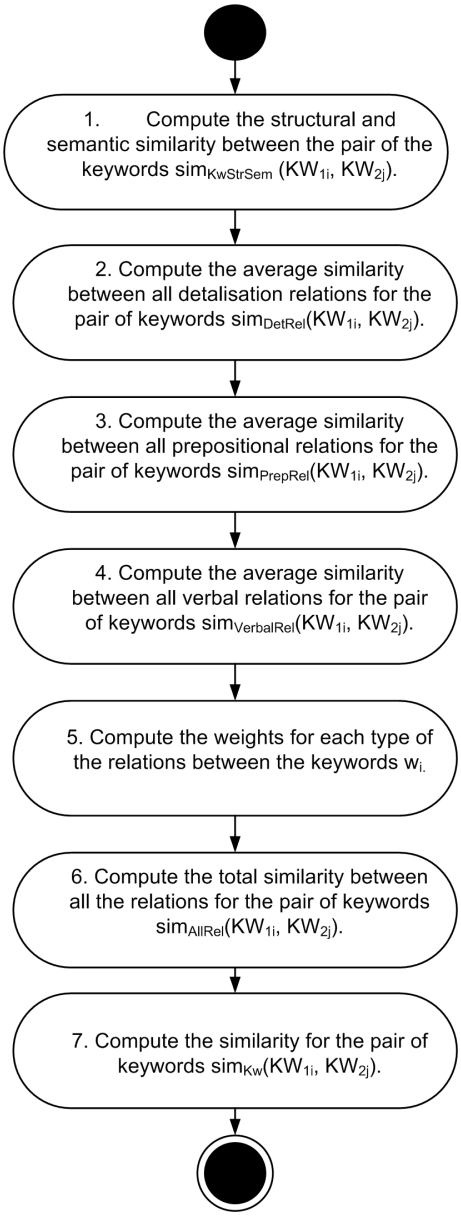

Figure 4.5: Comparison of a Pair of Keywords

Table 4.1: Comparison of the Keywords' Constituents

Kw_1 \ Kw_2	$[AdjDD_2]$	NDD_{2_1}	...	NDD_{2_n}
$[AdjDD_1]$	$sim_{wStrSem}$ $(AdjDD_1, AdjDD_2)$	×	×	×
NDD_{1_1}	×	$sim_{wStrSem}$ (NDD_{1_1}, NDD_{2_1})	... ...	$sim_{wStrSem}$ (NDD_{1_1}, NDD_{2_n})
...	×	...	...	...
NDD_{1_m}	×	$sim_{wStrSem}$ (NDD_{1_m}, NDD_{2_1})	... ...	$sim_{wStrSem}$ (NDD_{1_m}, NDD_{2_n})

semantic similarity measure for the pair of adjectives or nouns, which is calculated in accordance with Equation 4.5. The symbol ×means that the similarity value is not computed for a certain pair of words. It is used in the cells, which appear at the intersections of adjectives and nouns.

The structural and semantic similarity measure for the two keywords Kw_1 and Kw_2 is calculated in accordance with the following Formula.

$$sim_{Kw_{StrSem}}(Kw_1, Kw_2) = w_{Adj_{Kw_1Kw_2}} \times sim_{wStrSem}(AdjDD_{Kw_1}, AdjDD_{Kw_2}) + \\ + \ w_{NounKw_1Kw_2} \times sim_{NStrSemAvg}(NDD_{Kw_1}, NDD_{Kw_2}) \quad (4.6)$$

The weights for the Formula 4.6 are calculated in the following way.

$$w_{Adj_{Kw_1Kw_2}} = \frac{1}{Max(m,n)+1} \quad (4.7)$$

$$w_{Noun_{Kw_1Kw_2}} = 1 - w_{Adj_{Kw_1Kw_2}} \quad (4.8)$$

The intuition behind these formulae indicates, that they should reflect the importance of a single word as compared to all the others constituents of a keyword. For this reason, the weight for the adjectives is calculated by dividing 1 to the number of words in the longer keyword.

The average similarity value for the nouns of the first keyword with respect to the nouns of the second keyword (or vice versae) is computed in accordance with the greedy average, described in the section 4.2.1.

$$sim_{NStrSemAvg}(NDD_{Kw_1}, NDD_{Kw_2}) = \frac{\sum Max(Sim_{wStrSem}(NDD_1,\ NDD_2))}{Max(m,n)} \tag{4.9}$$

Comparison of the Relations

After having computed the structural and semantic similarity values for the pairs of keywords, the additional information on their likeness is received based on the comparison of their relations. This similarity considers only the relations between the whole keywords, which either explicitly belong to some topic of the module or appear as data domain objects of the learning outcomes.

In accordance with the data domain ontology described in the Section 3.5.2, three types of relations between a pair of keywords exist. They are *hasDetalisationRelation*, *hasPrepositionalRelation* and *hasVerbalRelation*.

Similarity Calculation for the Detalisation Relations

This is the second step of the algorithm presented in the Figure 4.5.

In the data domain ontology the object property *hasDetalisationRelation* has the following subproperties: *suchAs*, *e.g.*, *forExample*, *forInstance* and some others. All of these phrases are synonymous. When used in the natural language text between two collocations, each of these expressions exemplify the first phrase by the means of the second. It means that the object from the first words' combination generalizes the objects, which follow. For this reason the similarity measure for this relation considers two similarities: between the domains and the ranges of the properties. The similarity value between the two detalising relations is computed as the average between the similarities for its domain and range.

$$\begin{aligned} sim_{DetRel}(Kw_1, Kw_2) = 0.5 \times sim_{Kw_{StrSem}}(RelDom_{Kw_1}, RelDom_{Kw_2}) + \\ + 0.5 \times sim_{DetRelRanAvg}(RelRange_{Kw_1}, RelRange_{Kw_2}) \end{aligned} \tag{4.10}$$

Table 4.2: Comparison of the Relations' Ranges

$RelRange_{Kw_2}$ / $RelRange_{Kw_1}$	$RelRange_{Kw_{2_1}}$	$RelRange_{Kw_{2_n}}$
$RelRange_{Kw_{1_1}}$	$sim_{Kw_{StrSem}}(RelRange_{Kw_{1_1}}, RelRange_{Kw_{2_1}})$	$sim_{Kw_{StrSem}}(RelRange_{Kw_{1_1}}, RelRange_{Kw_{2_n}})$
$RelRange_{Kw_{1_m}}$	$sim_{Kw_{StrSem}}(RelRange_{Kw_{1_m}}, RelRange_{Kw_{2_1}})$	$sim_{Kw_{StrSem}}(RelRange_{Kw_{1_m}}, RelRange_{Kw_{2_n}})$

We assert, that the domain consists of one keyword, while the range can be presented by a sequence of keywords, for example, "... relational database management systems, such as Oracle and MySQL". The similarity between the domains of the relation is calculated in accordance with algorithm for computing structural and semantic similarity measure, which was described in subsection 4.2.2. In general, the similarity value between the ranges is calculated in the following way. Firstly, the structural and semantic similarities are computed for each pair of the keywords from the ranges. The following matrix is filled in with the resulting numbers.

The similarity between each pair of the keywords, representing the ranges, $Sim_{w_{StrSem}}(RelRange_{Kw_{1_i}}, RelRange_{Kw_{2_j}})$ is computed in accordance with the algorithm presented in the section 4.2.2.

The values in the matrix 4.2 are processed using the greedy iterative algorithm, described in the section 4.2.1. The resulting value of $sim_{RelRanAvg}(RelRange_{Kw_1}, RelRange_{Kw_2}$ is received using the following Formula.

$$sim_{DetRelRanAvg}(RelRange_{Kw_1}, RelRange_{Kw_2}) = \frac{\sum Max(Sim_{Kw_{StrSem}}(RelRange_{Kw_1}, RelRange_{Kw_2}))}{Max(m,n)} \tag{4.11}$$

Similarity Calculation for the Prepositional Relations

This Section describes the step 3 of the algorithm presented in the Figure 4.5. The similarity measure for all the prepositional relations, which link the domain keyword to the range (-s), is calculated in the following way.

Table 4.3: Comparison of the Subrelations' Ranges

$SubRel_{Kw_2}$ / $SubRel_{Kw_1}$	$SubRel_{1_{Kw_2}}$	$SubRel_{n_{Kw_2}}$
$SubRel_{1_{Kw_1}}$	$sim_{Kw_{StrSem}}$ ($SubRel_1Range_{Kw_1}$, $SubRel_1Range_{Kw_2}$)	$sim_{Kw_{StrSem}}$ ($SubRel_1Range_{Kw_1}$, $SubRel_nRange_{Kw_2}$)
$SubRel_{m_{Kw_1}}$	$sim_{Kw_{StrSem}}$ ($SubRel_mRange_{Kw_1}$, $SubRel_1Range_{Kw_2}$)	$sim_{Kw_{StrSem}}$ ($SubRel_mRange_{Kw_1}$, $SubRel_nRange_{Kw_2}$)

$$sim_{PrepRel}(Kw_1, Kw_2) = 0.5 \times sim_{Kw_{StrSem}}(RelDom_{Kw_1}, RelDom_{Kw_2}) + \\ + 0.5 \times sim_{PrepRelRanAvg}(RelRange_{Kw_1}, RelRange_{Kw_2}) \quad (4.12)$$

The *hasPrepositionalRelation* has prepositions as its subproperties. The similarity measure for the pair of prepositional relations is calculated in accordance with the following algorithm. The matrix is created, which contains as many rows as the first keyword's Kw_1 prepositional subproperties (they include, for example, "of", "in" etc.). If a keyword is related to several ranges through one subrelation, a separate row for each of them is created. The columns are received analogically, but based on the prepositional subrelations of the second keyword Kw_2.

The similarities in the Table 4.3 are computed only for the ranges, which are connected by the same prepositions.

$$sim_{Kw_{StrSem}}(SubRel_iRange_{Kw_1}, SubRel_jRange_{Kw_2}) = \begin{cases} 0, \\ \text{if the ranges belong to different} \\ \text{subrelations} \\ sim_{Kw_{StrSem}}(SubRel_iRange_{Kw_1}, \\ SubRel_jRange_{Kw_2}) \\ \text{in other case.} \end{cases} \quad (4.13)$$

The empty rows and columns are eliminated from the Table 4.3, the numbers m and n are reduced by the number of rows and columns removed.

$$\begin{aligned} m' &= m - NumNullRows \\ n' &= n - NumNullColumns \end{aligned} \tag{4.14}$$

The average value, which characterises the similarity between the ranges of the prepositional subrelations of the keywords Kw_1 and Kw_2, is calculated in accordance with the greedy iterative strategy described in Section 4.2.1.

$$\begin{aligned} sim_{PrepRelRanAvg}(RelRange_{Kw_1}, RelRange_{Kw_2}) = \\ \frac{\sum Max(sim_{Kw_{StrSem}}(SubRelRange_{Kw_1}, SubRelRange_{Kw_1}))}{Max(m', n')} \end{aligned} \tag{4.15}$$

Similarity Calculation for the Verbal Relations

In this section we represent the fourth step of the algorithm presented in the Figure 4.5. The *hasVerbalRelation* introduces the relations, which link the data domain keywords through the verbs. Each particular verb is stored as a subproperty of the *hasVerbalRelation*. The similarity measure for the verbal relations contains three constituents: the similarity values between domains, ranges and the verbs representing the relations. The similarity measure for all the verbal relations of a keyword is calculated in the following way.

$$\begin{aligned} sim_{VerbalRel}(Kw_1, Kw_2) = 0.5 \times sim_{Kw_{StrSem}}(RelDom_{Kw_1}, RelDom_{Kw_2}) + \\ + 0.5 \times sim_{VerbRelRanAvg}(VerbRelRange_{Kw_1}, VerbRelRange_{Kw_2}) \end{aligned} \tag{4.16}$$

For a pair of keywords, having verbal relations, firstly, the structural and semantic similarity between themselves is computed. Afterwards for each of them all the verbal relations and domains are extracted. The same similarity measure is calculated for each of the pairs.

The *hasVerbalRelation* has verbs as its subproperties. The similarity measure for the pair of verbal relations is calculated in accordance with the following algorithm. The matrix is created, which contains as many rows as the first keyword's Kw_1

Table 4.4: Comparison of the Verbal Subrelations and Ranges

$VerbRelRange_{Kw_2}$ / $VerbRelRange_{Kw_1}$	$VerbRelRange_{1_{Kw_2}}$	$VerbRelRange_{n_{Kw_2}}$
$VerbRelRange_{1_{Kw_1}}$	$simAvg_{Kw_{StrSem}}$ ($VerbRelRange_{1_{Kw_1}}$, $VerbRelRange_{1_{Kw_2}}$)	$simAvg_{Kw_{StrSem}}$ ($VerbRelRange_{1_{Kw_1}}$, $VerbRelRange_{n_{Kw_2}}$)
$VerbRelRange_{m_{Kw_1}}$	$simAvg_{Kw_{StrSem}}$ $VerbRelRange_{m_{Kw_1}}$, $VerbRelRange_{1_{Kw_2}}$)	$simAvg_{Kw_{StrSem}}$ $VerbRelRange_{m_{Kw_1}}$, $VerbRelRange_{n_{Kw_2}}$)

verbal subproperties. If a keyword is related to several ranges through one subrelation, a separate row for each of them is created. The columns are received analogically, but based on the prepositional subrelations of the second keyword Kw_2.

Each value in the Table 4.4 is received using the following Equation.

$$\begin{aligned} simAvg_{Kw_{StrSem}}(VerbRelRange_{i_{Kw_1}}, VerbRelRange_{j_{Kw_2}}) = \\ 0.5 \times sim_{Kw_{StrSem}}(Verb_{i_{Kw_1}}, Verb_{j_{Kw_2}}) + \\ +0.5 \times sim_{Kw_{StrSem}}(RelRange_{i_{Kw_1}}, RelRange_{j_{Kw_2}}) \end{aligned} \tag{4.17}$$

The average value, which characterizes the similarity between the verbs and ranges of the verbal subrelations of the keywords Kw_1 and Kw_2, is calculated in accordance with the greedy iterative strategy described in Section 4.2.1.

$$\begin{aligned} sim_{VerbRelRanAvg}(VerbRelRange_{Kw_1}, VerbRelRange_{Kw_2}) = \\ \frac{\sum Max(sim_{Kw_{StrSem}}(VerbRelRange_{Kw_1}, VerbRelRange_{Kw_1}))}{Max(m,n)} \end{aligned} \tag{4.18}$$

The Similarity Measure for the Pair of Keywords

The similarity between a pair of keywords is computed in accordance with the following Formula.

$$sim_{Kw}(KW_{1_i}, KW_{2_j}) = \begin{cases} 0.5 \times sim_{Kw_{StrSem}}(KW_{1_i}, KW_{2_j}) + \\ 0.5 \times sim_{AllRel}(KW_{1_i}, KW_{2_j}) \\ \text{if the relations were compared;} \\ sim_{Kw_{StrSem}}(KW_{1_i}, KW_{2_j}) \text{ otherwise.} \end{cases} \tag{4.19}$$

Here $sim_{AllRel}(KW_{1_i}, KW_{2_j})$ denotes the similarity for all the relations between the two keywords.

$$\begin{aligned} sim_{AllRel}(KW_{1_i}, KW_{2_j}) = w_1 \times sim_{DetRel}(Kw_{1_i}, Kw_{2_j}) + \\ + w_2 \times sim_{PrepRel}(Kw_{1_i}, Kw_{2_j}) + \\ + w_3 \times sim_{VerbalRel}(Kw_{1_i}, Kw_{2_j}) \end{aligned}$$

In this work we utilized the equal weights ($w_i = \frac{1}{3}$ if all relations were compared, $w_i = \frac{1}{2}$ if two types of relations were compared, $w_i = 1$ if one type of relation was compared).

4.3.2 Comparison of the Keywords through the Topics

The total similarity for a pair of keywords considers the topics, to which they belong by computing the aggregated similarity for all the sets of their keywords.

The algorithm for comparison of the keywords of the topics, to which the compared pair of keywords belongs, is presented in Figure 4.6.

The algorithm contains the following steps.

At first the matrix for all the keywords, belonging to the first and second topics, is built (steps 1-2 in the Figure 4.6). Each cell of the Table 4.5 contains the similarity value between the two keywords Kw_1 and Kw_2, which is calculated in accordance with the Formula 4.19.

The average value $sim_{KwByTop}$ for the Matrix 4.5 is computed in accordance with the maximum average strategy, where threshold τ is set to zero (step 3 in Figure 4.6).

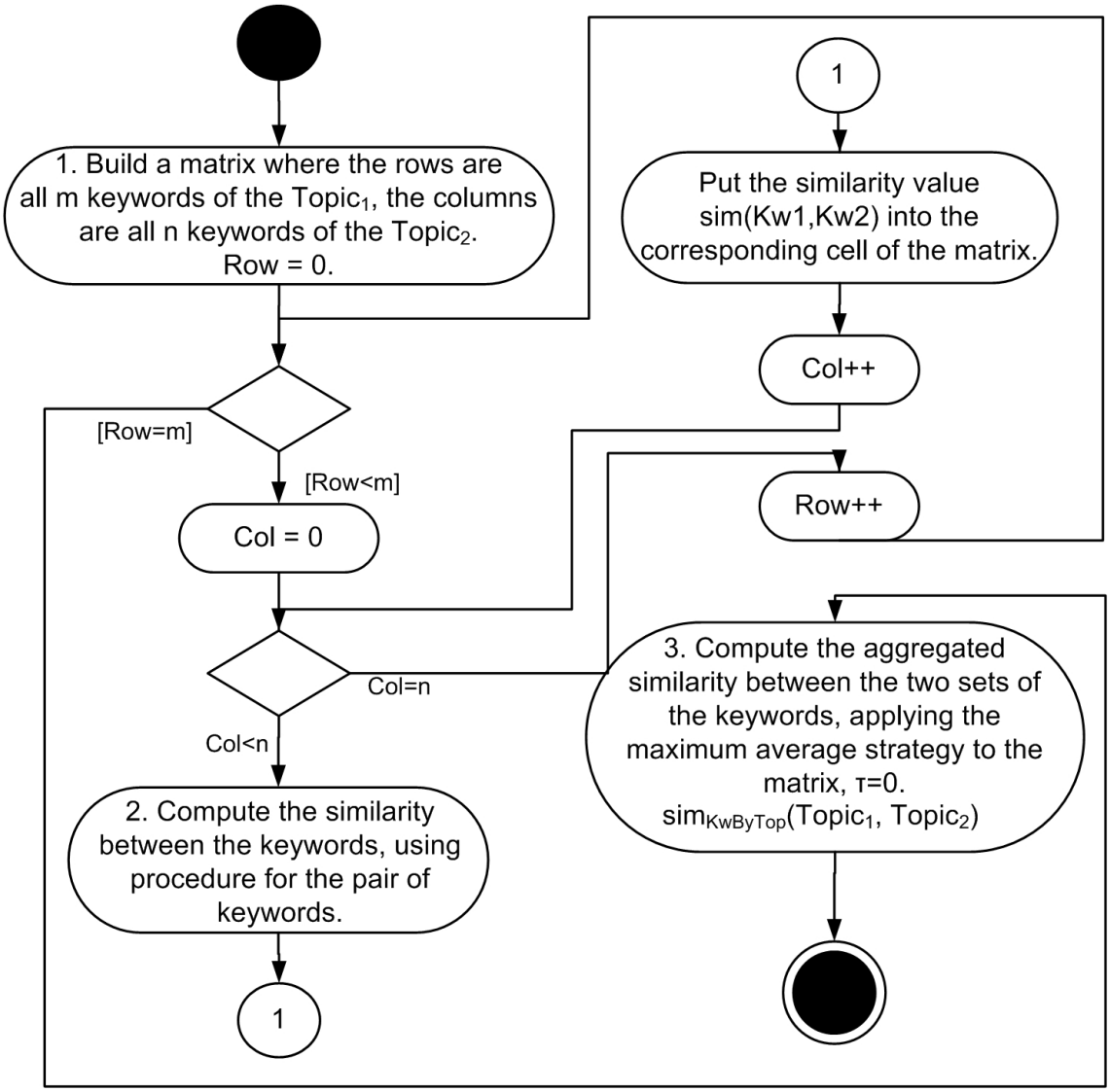

Figure 4.6: Comparison of the Keywords through the Topics

Table 4.5: Comparison of the Keywords of Topics

Kw_1 \ Kw_2	Kw_{2_1}	Kw_{2_n}
Kw_{1_1}	$sim_{Kw}(KW_{1_1}, KW_{2_1})$	$sim_{Kw}(KW_{1_1}, KW_{2_n})$
Kw_{1_m}	$sim_{Kw}(KW_{1_m}, KW_{2_1})$	$sim_{Kw}(KW_{1_m}, KW_{2_n})$

4.3.3 The Total Similarity for a Pair of Keywords

The total similarity value for the two keywords is computed as an average between sim_{Kw} and $sim_{KwByTop}$.

$$sim_{Kw_{tot}} = \frac{sim_{Kw} + sim_{KwByTop}}{2} \tag{4.20}$$

4.3.4 Aggregation of the Similarities between the Keywords

In order to compute the similarity between the sets of keywords of the two educational modules (or courses), we built the matrix (similar to the 4.5), where the cells contain the similarities between all the keywords and topics of both modules. The similarity for each pair of keywords is computed in accordance with Formula 4.20.

Afterwards the following algorithm is applied.

1. Set the threshold to eliminate all the values, which show, that two keywords are different. In this work we set the threshold to the value $\tau = 0.40$ (step 1 in the algorithm presented in Figure 4.4).
2. Remove all the values from the matrix, which are below the threshold value. We did not remove the empty rows and columns in order to consider the dissimilarity between the bags of keywords based on their total number.
3. Calculate the similarity $sim_{Kw_{aggr}}$ in accordance with the maximum average presented in Section 4.2.1 (step 6 in the algorithm presented in the Figure 4.4).

4.4 Similarity Measure for Comparison of the Action Verbs

In this work we introduce a semantic measure for comparison of the action verbs. We called our measure "CAVe", because it reflects the **c**ognitive space's semantic

similarity between the **a**ction **ve**rbs. The grounds for the decision to create a novel similarity measure lie in the hypothesis that when comparing the actions, expressed by the verbs, an expert pays respect not only to their particular sense, but also evaluates them at a more general level of the type of activity. We suppose that diverse verbs used in the LOs can be substituted by the names of the revised taxonomy's categories and subcategories and preserve the general sense of the learning outcomes, because they will still express the action of the same type, even if the verbs' and categories' titles are not synonymous.

4.4.1 Adjustments to D. Krathwohl's Taxonomy

Considering the analysis of B. Bloom's taxonomy of educational objectives revised by D. Krathwohl, we based the similarity measure for comparison of the action verbs on the distances in Cognitive Process dimension. We decided to convert the taxonomy of the Cognitive Process dimension to a two-dimensional space and assign coordinates to its categories and subcategories in order to create a similarity measure, based on the distance between the verbs in this model.

D. Krathwohl's revised version of B. Bloom's taxonomy is presented in Appendix A. Table 4.6 presents the two-dimensional model for storing the action verbs, which we created from the Cognitive Process dimension of the taxonomy. The columns in Table 4.6 represent the categories of the taxonomy, while the rows stand for the subcategories.

We put the categories of the taxonomy along the horizontal axis, called the axis in the same way as in D. Krathwohl's variant of taxonomy and assigned to each coordinates from 1 to 6. We spread the subcategories along the vertical axis, called the axis "Complexity" and assigned coordinates to each of the verbs. However, the process was not that linear as it was for the categories. The intuition is that the model contains the actions of approximately the same complexity belonging to different Cognitive Process dimensions. If we applied linear sequence, we should have admitted, for example, that to **execute** an action with respect to Application is more complicated, than to **explain** a notion with respect to category Understand and so on. If we took another strategy and started giving coordinates to subcategories from 1 inside each category, we would have faced the same problems. Therefore, we found solution in spreading the subcategories along the

Table 4.6: Two-dimensional Representation of the Cognitive Space of the Revised Taxonomy of Educational Aims

CPD / Complexity	Remember 1	Understand 2	Apply 3	Analyze 4	Evaluate 5	Create 6
1	Recognize					
2	Recall					
3		Interpret	Execute			
4		Exemplify		Differentiate	Check	
5		Classify		Organize		
6		Summarize		Attribute		
7		Infer				
8		Compare				
9		Explain	Implement			
10					Critique	
11						Generate
12						Plan
13						Produce

"Complexity" axis, as if they did not belong to the categories. This means, that actions of approximately the same complexity belonging to different Cognitive Process dimensions have the same Complexity coordinates. The gaps between two verbs inside one category appear when another category holds the activities, which are more complex than the first and less complex than the second from the first category. This method led to construction of the following Table 4.6. In Table 4.6 "CPD" stands for Cognitive Process dimension, while "Complexity" is the axis for the subcategories. The numbers denote the coordinates of the verbs along the corresponding axes. The complexities were defined in accordance with the author's opinion. However, they can be assigned in a different way by another expert. This will not influence the algorithm for computing the similarity measure for the action verbs. The algorithm is depicted in detail in the following sections.

4.4.2 Semantic Similarity Measure CAVe

The measure CAVe is based on calculation of the distances between action verbs in the revised taxonomy by D. Krathwohl.

$$sim_{CAVe}(AV_i, AV_j) = \frac{max(d_{EucKrth}(AV_i, AV_j)) - cur(d_{EucKrth}(AV_i, AV_j)}{max(d_{EucKrth}(AV_i, AV_j))}$$
$$sim_{CAVe}(AV_i, AV_j) \in [0, 1]. \quad (4.21)$$

In Equation 4.21 $cur(d_{EucKrth}(AV_i, AV_j))$ represents the distance between the subcategories, to which the action verbs AV_i and AV_j refer. The $max(d_{EucKrth}(AV_i, AV_j))$ stands for the longest possible distance in the model.

The CAVe similarity preserves all the main properties, which apply to any similarity measure:

1. sim_{CAVe} always takes non-negative value, because any possible distance between the action verbs will be less or equal to maximum possible distance in the model.
2. The similarity between the verbs belonging to the same subcategories equals 1, as in this case $cur(d_{EucKrth}(AV_i, AV_j)) = 0$.
3. sim_{CAVe} is symmetric as it is based on distances, which do not depend on the direction of comparison.

Distance in Cognitive Space

We assign two coordinates to an action verb in the cognitive space as they appear in the Table 4.6. Therefore each action verb AV is represented as the following vector.

$$AV = (CPD; Compl) \quad (4.22)$$

In Equation 4.22 *CPD* stands for the coordinate of an action verb in the cognitive process dimension (category of the revised taxonomy), while *Compl* characterizes complexity inside the CPD. CAVe utilizes the euclidean distance between the action verbs in the two-dimensional cognitive space, based on the action verbs' coordinates in the two-dimensional cognitive space of D. Krathwohl's model.

In order to calculate the distance between two action verbs we must consider that the distance between any neighbouring categories must exceed the longest possible distance between subcategories inside any category. The intuition behind this statement is that the Complexity coordinates characterize different subcategories inside one type of cognitive activity. This requirement implies normalization of the axes. For this purpose we introduced the coefficient w to normalize the Cognitive Process axis with respect to the Complexity axis. In this case the euclidean distance between the action verbs in D. Krathwohl's taxonomy will be calculated in accordance with the following Equation.

$$d_{EucKrth}(AV_i, AV_j) = \sqrt{w^2 \times (CPD_i - CPD_j)^2 + (Compl_i - Compl_j)^2} \tag{4.23}$$

In Formula 4.23 the abbreviation AV stands for the action verb, CPD stands for the cognitive process dimension coordinate, while $Compl$ reflects the coordinate along the Complexity axis.

Using Formula 4.23 for the distance, we can formally express the rule regarding normalisation of axes and thus compute the exact value of the coefficient w through the following inequality.

$$\sqrt{w^2 \times (CPD_m - CPD_{m+1})^2 + (min|Compl_m - Compl_{m+1}|)^2} > \sqrt{0 + (max|Compl_{max} - Compl_{min}|)^2} \tag{4.24}$$

In Inequality 4.24 CPD_m and CPD_{m+1} represent the neighboring categories in the cognitive process dimension, while $Compl_m$ and $Compl_{m+1}$ reflect the complexity subcategories in CPD_m and CPD_{m+1} respectively.

The expression in the left side of Inequality 4.24 is used to identify the shortest possible distance between categories in the model. In order to calculate it, we must compute distances between all the subcategories of the neighbouring categories (CPD_m and CPD_{m+1}) and find the minimum ($min|Compl_m - Compl_{m+1}|$). The distance between any neighbouring categories equals 1. Table 4.7 represents the minimal distances between subcateogries for each pair of neighbouring categories. The minimal value ($min|Compl_m - Compl_{m+1}|$) equals zero.

Table 4.7: The Minimal Distances between Subcategories of Neighbouring Categories

Neighbouring Categories	Remember/ Understand	Understand/ Apply	Apply/ Analyze	Analyze/ Evaluate	Evaluate/ Create
Minimal distance between subcategories	$\|2-3\|=1$	$\|3-3\|=0$ $\|9-9\|=0$	$\|3-4\|=1$	$\|4-4\|=0$	$\|10-11\|=1$

Table 4.8: The Maximal Distances between Subcategories of Each Category

Categories	Remember	Understand	Apply	Analyze	Evaluate/	Create
Maximal distance	2-1=1	**9-3=6**	**9-3=6**	6-4=2	**10-4=6**	13-11=2

The right hand side expression in inequality 4.24 symbolizes the longest distance between the subcategories inside one category (the zero represents one category). In order to compute it, we calculate the difference between the maximal and minimal coordinates of subcategories inside each category $|Compl_{max} - Compl_{min}|$ (see Table 4.8) and afterwards find the maximum value. According to Table 4.8, $max|Compl_{max} - Compl_{min}| = 6$.

When we substitute the expressions in the Inequality 4.24 with their computed values, we receive the following revised Formula.

$$\sqrt{(w^2 \times 1)^2 + (0)^2} > \sqrt{0 + (6)^2} \tag{4.25}$$

From Inequality 4.25 we imply, that the coefficient $w > 6$. Therefore, the minimal possible integer value is 7. In our work we utilised the normalising coefficient $w = 7$.

4.4.3 Sub-Ontology of Verbs

In order to store the adjusted D. Krathwohl's taxonomy we created a special ontology, which expands the basic course and module ontology by the following three classes. The sub-ontology of verbs is presented in Figure 4.7.

1. **ActionVerb** is the class, used to unite the representation of D. Krathwohl's revised taxonomy. It is the superclass for all the other classes in this small ontology.

2. **ClusterAV** contains the verbs from the cognitive process dimension of the D. Krathwohl's revised taxonomy. The individuals of this class have the data properties *ClAVCPD* and *ClAVComplexity*, which identify them in the cognitive process and complexity dimensions. The first value is taken from the model presented in table 4.6, while the second is forced to zero value.

3. **SubClusterAV** contains the verbs, symbolising complexities of the D. Krathwohl's adjusted model. Each instance from this class is linked through the relation *isSubclusterInCluster* to one of the individuals of the class **ClusterAV**. The relations are set in accordance with the adjusted taxonomy. The individuals of this class have the data properties *SubClAVCPD* and *SubClAVComplexity*, which identify their coordinates in the cognitive process and complexity dimensions. The first value is inferred from the corresponding instances of the **ClusterAV** class. The latter coordinate is taken from the model presented in table 4.6.

4. **DataDomainActionVerb** is used to store the verbs, which are used in the learning outcomes of specific data domains and are not included in the taxonomy. Each instance of this class is linked through *isDDAVInSubcluster* object property to an individual of the class **SubClusterAV**. These links are set by the experts in accordance with their opinion. The coordinates of the data domain verbs in the cognitive process and complexity dimensions are inherited from the corresponding individuals of the **SubClusterAV** class.

The ontology also contains the SWRL-rules, which are used to infer the coordinates of the subcluster and data domain action verbs based on their attachment to cluster and subcluster verbs respectively.

In practice, the lifecycle of the ontology of verbs contains the following stages. The verbs from the taxonomy of D. Krathwohl are once manually input into the above described structure. Afterwards this ontology is enriched with the data domain verbs by an expert, who sets the links needed for inference of their coordinates. The expert can adjust the model in the part of data domain verbs in accordance with his own understanding. He can set and reset the relations *isInSubClusterAV*

between the individuals of the classes **SubClusterAV** and **DataDomainActionVerb**, bearing in mind, that this affects the results of comparison of the learning outcomes, which contain them. This is due to the fact, that after all the relations are set, the reasoner runs and infers the coordinates for the data domain action verbs in accordance with their ”subclusters”.

4.4.4 Combination of the Ontologies

In Figure 4.8 we illustrate combination of the course, module, learning outcome, data domain and verbs' ontologies. The class **ActionVerb** (from the sub-ontology of the learning outcomes) is equal to the class with the same from the ontology of verbs.

4.5 Comparison of the Learning Outcomes

Comparison of the learning outcomes is based on alignment of their ontologies and contains the following major stages as presented in the Figure 4.9. Here we will be using the titles of the ontology entities to refer to the learning outcomes' constituents.

1. Set the threshold $\tau DDOC$ for data domain object constructions, which is used to eliminate those with low similarity values. In this work we utilised $\tau = 0.50$. Build a matrix with the rows and columns representing the learning outcomes of the Module M_1 (Course C_1) and Module 2 (Course C_2) respectively (step 1 in the algorithm in the Figure 4.9).

2. Compare pairwise all the learning outcomes LO_1 and LO_2 of educational modules M_1 and M_2 (or educational courses C_1 and C_2) (step 2 in the algorithm in the Figure 4.9).

3. Aggregate the similarity values between the pairs of the learning outcomes in order to receive the total LOs' similarity for the educational modules or courses using the maximum average strategy (step 3 in the algorithm in Figure 4.9).

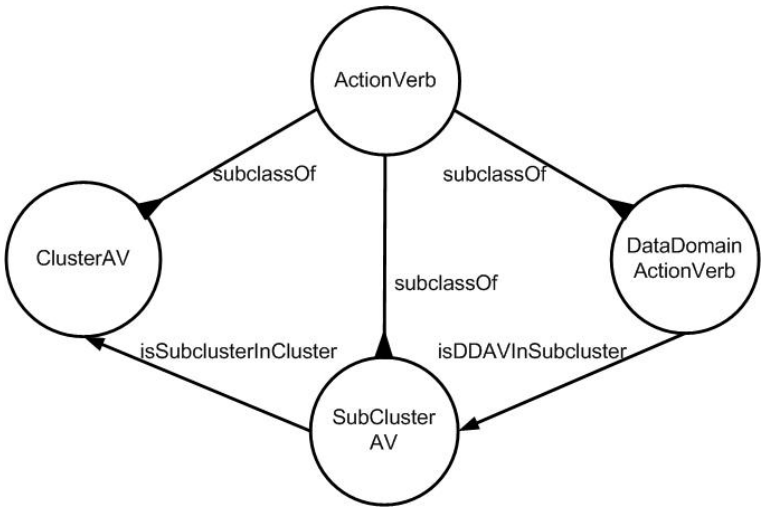

Figure 4.7: The Ontology of Verbs

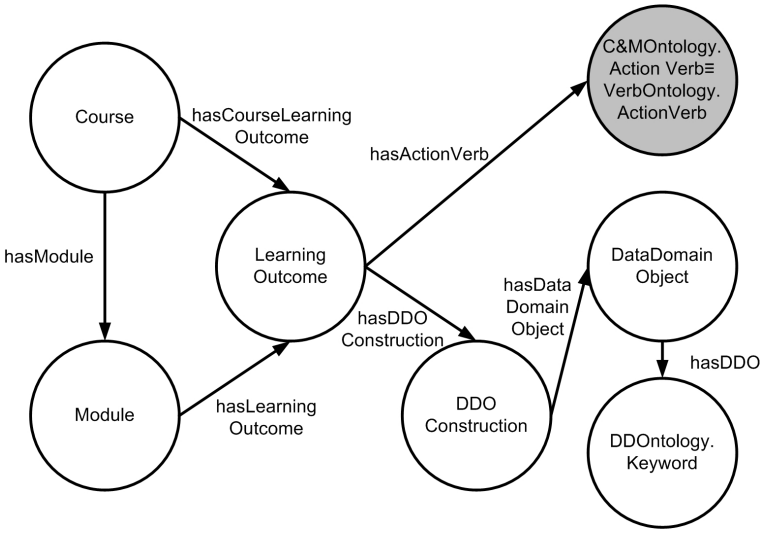

Figure 4.8: Combination of the Course, Module, Learning Outcome, Data Domain and Verbs' Ontologies

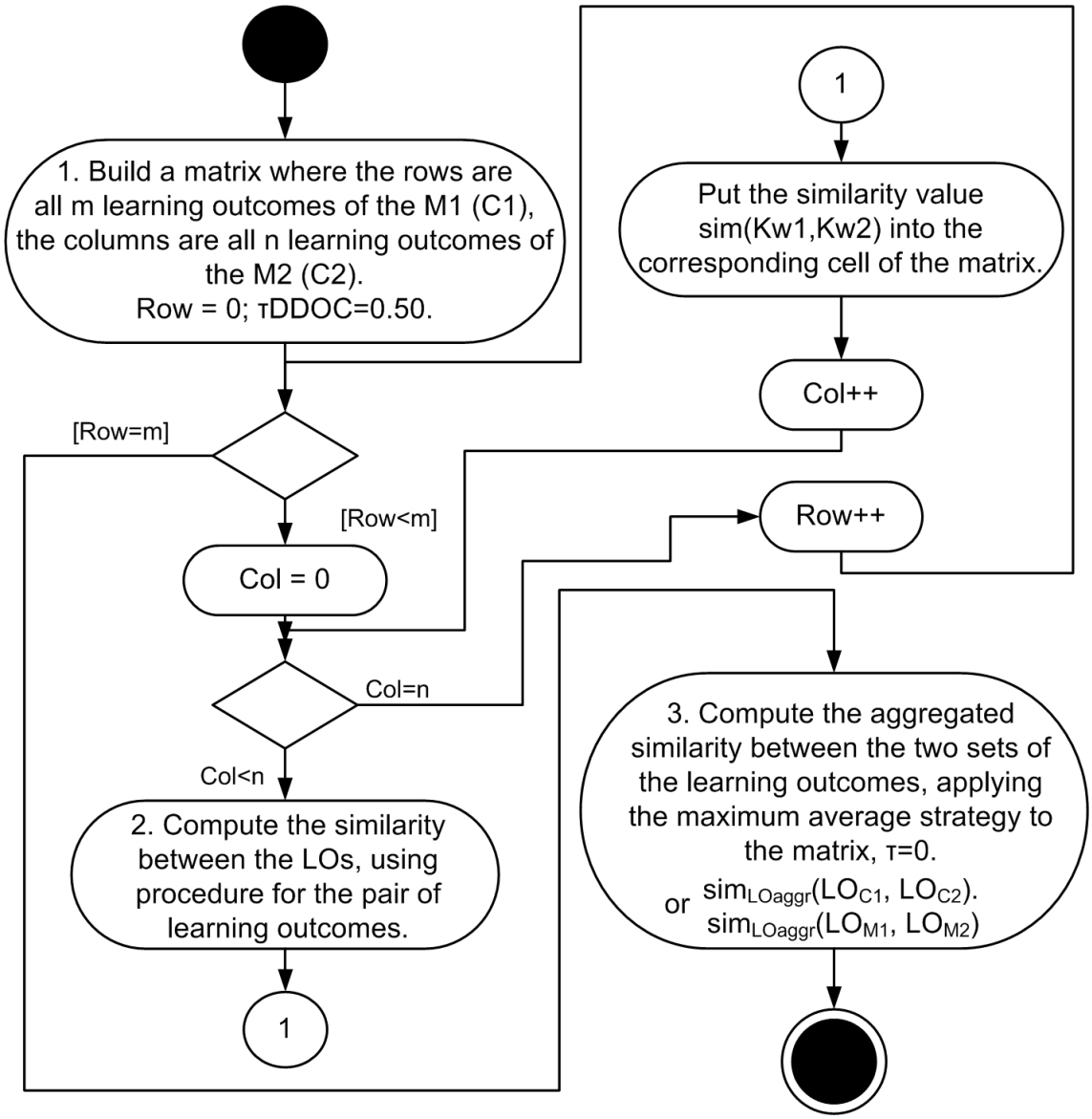

Figure 4.9: The Alignment of the Learning Outcomes of Modules (Courses)

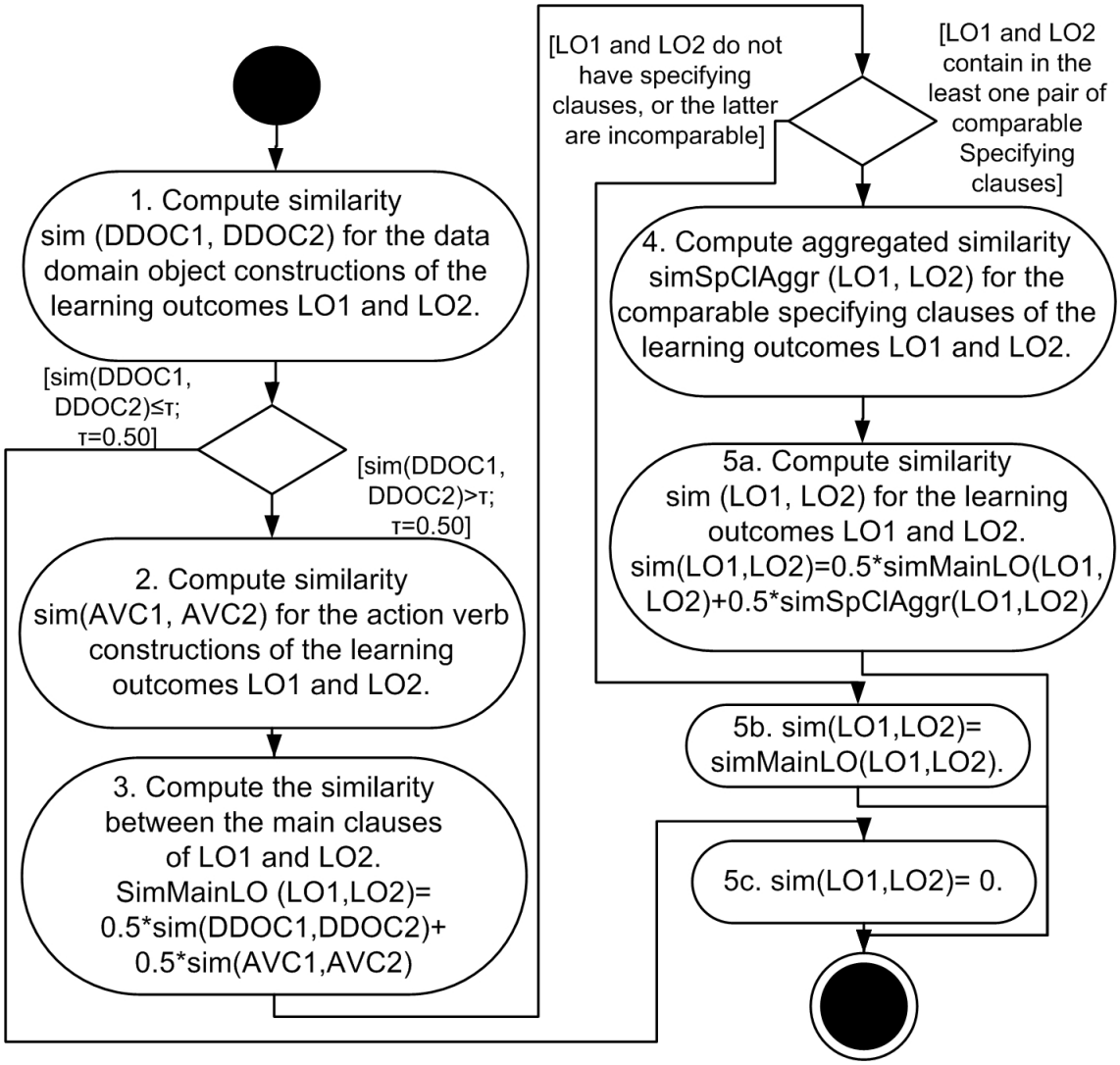

Figure 4.10: Comparison of a Pair of Learning Outcomes

4.5.1 Comparison of a Pair of Learning Outcomes

For each pair of the learning outcomes do the following (the number of step in the list below coincides with the number in Figure 4.10, presenting the algorithm).

1. Compare data domain object constructions $DDOC_{1_i}$ and $DDOC_{2_j}$. Set the similarity between the learning outcomes to zero, if the data domain object constructions produced similarity below or equal to the threshold's value.
2. If the similarity between the data domain object constructions was greater than the threshold, compute similarity value for the action verb constructions AVC_{1_i} and AVC_{2_j}.
3. Compute the similarity value for the main parts of the learning outcomes, based on the values received at the previous steps. The similarities between the main parts are computed as average for the similarities between data domain and action verbs constructions.

$$sim_{MainLO} = 0.5 \times sim_{DDOC}(DDOC_1, DDOC_2) + 0.5 \times sim_{AVC}(AVC_1, AVC_2) \tag{4.26}$$

4. Compute the aggregated similarity value for the specifying clauses of the learning outcomes, if each of the learning outcomes contains in the least one such clause. A similarity measure for a pair of specifying clauses is calculated only if one of the following conditions holds:
   (a) both specifying clauses start with a verb or a participle;
   (b) both specifying clauses start with the same specifying conjunction.

   Compute the average value for the specifying clauses of the outcomes in accordance with the greedy iterative strategy.
5. Compute the total similarity between the pair of learning outcomes.

The similarity between two learning outcomes considers the similarities between its main parts and specifying clauses, if they satisfy the conditions set above. The

Table 4.9: Comparison of the Data Domain Objects

DDO_{DDOC_2} / DDO_{DDOC_1}	DDO_{2_1}	DDO_{2n}
DDO_{1_1}	$sim_{DDO}(DDO_{1_1}, DDO_{2_1})$	$sim_{DDO}(DDO_{1_1}, DDO_{2_n})$
DDO_{1_1}	$sim_{DDO}(DDO_{1_m}, DDO_{2_1})$	$sim_{DDO}(DDO_{1_m}, DDO_{2_n})$

main parts and specifying clauses are treated as equally important by setting the same weights.

$$sim_{LO}(LO_1, LO_2) = \begin{cases} 0.5 \times sim_{MainLO}(LO_1, LO_2) + 0.5 \times sim_{SpCl_{aggr}}(LO_1, LO_2) \\ \text{,if the specifying clauses are compared} \\ sim_{MainLO}(LO_1, LO_2) \text{ otherwise.} \end{cases} \tag{4.27}$$

Comparison of the Data Domain Object Constructions

In accordance with the ontological model, the data domain object construction contains one or more data domain objects, whereas each of them consists of one or several keywords. The similarity between data domain object constructions is computed in accordance with the greedy iterative strategy, using the matrix of similarities between the corresponding data domain objects (see Table 4.9).

The similarity between a pair of data domain objects is computed in accordance with the strategy described in Section 4.3.3 for the pairs of keywords.

$$sim_{DDO}(DDO1_1, DDO_2) = sim_{Kw_{tot}}(KW_{DDO_1}, KW_{DDO_2})$$

$$sim_{DDOC}(DDOC_1, DDOC_2) = \frac{\sum Max(sim_{DDO}(DDO_{DDOC_1}, DDO_{DDOC_2}))}{Max(m, n)} \tag{4.28}$$

The similarity between a pair of data domain object constructions is calculated in accordance with the greedy iterative strategy, using the matrix of similarities between the corresponding keywords. The similarity between a pair of keywords is computed in conformity with the algorithm described in Section 4.3.1.

Comparison of the Action Verb Constructions

An action verb construction may contain from one to several action verbs of a learning outcome.

We assume, that the usage of a combined similarity measure, which includes the similarity computed between the verbs as between the words ($sim_{wStrSem}(w_1, w_2)$) and CAVe will improve the quality of comparison of the action verbs and smoothen the inaccuracies brought by each of the approaches when used on their own.

$$sim_{AV} = 0.5 \times sim_{wStrSem}(AV_1, AV_2) + 0.5 \times sim_{CAVe}(AV_1, AV_2) \tag{4.29}$$

The similarities between each pair of action verbs are gathered into a matrix, afterwards the average is computed in accordance with the greedy iterative strategy, in the same manner as for the data domain object constructions.

$$sim_{AVC}(AVC_1, AVC_2) = \frac{\sum Max(sim_{AV}(AV_{AVC_1}, AV_{AVC_2}))}{Max(m, n)} \tag{4.30}$$

Comparison of the Specifying Clauses

The algorithm for comparison of the specifying clauses is presented in the Figure 4.11 and contains the following steps.

Firstly, a matrix 4.10 for similarities between the specifying clauses is built (step 1 in Figure 4.11).

The similarity for a pair of specifying clauses is calculated in accordance with the following Formula.

$$sim_{SpCl}(SCl_1, SCl_2) = \begin{cases} sim_{DDOC}(DDOC_{SCl_1}, DDOC_{SCl_2}), \text{if the clauses contain} \\ \text{only data domain object constructions} \\ sim_{Tail}(SCl_1, SCl_2) \text{ ,if the clauses contain verbal} \\ \text{and data domain object constructions.} \end{cases} \tag{4.31}$$

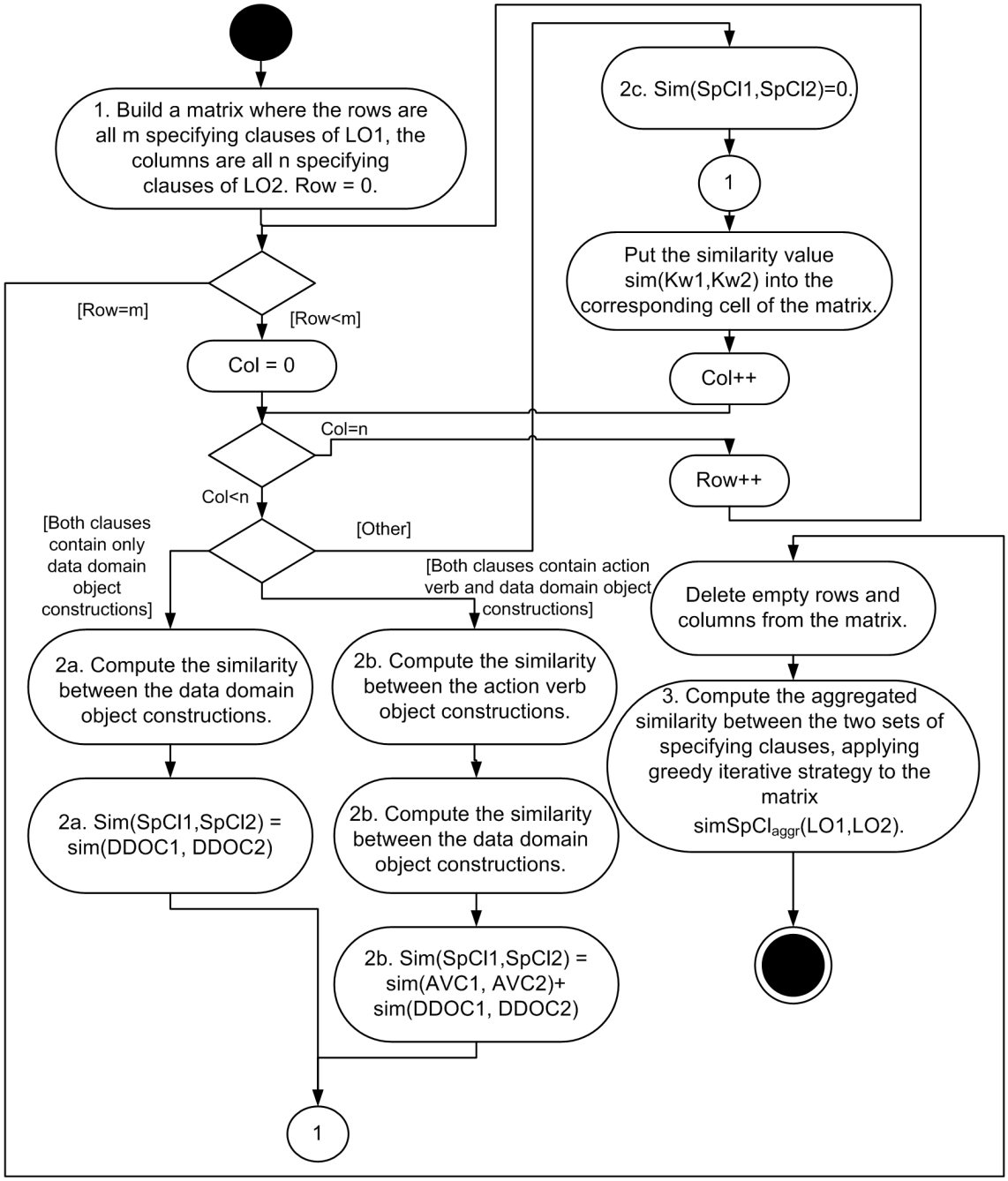

Figure 4.11: Comparison of the Specifying Clauses

Table 4.10: Comparison of the Specifying Clauses

$SpCl_{LO_1}$ \ $SpCl_{LO_2}$	$SpCl_{1_{LO_2}}$	$SpCl_{n_{LO_2}}$
$SpCl_{1_{LO_1}}$	$sim_{SpCl}(SpCl_{1LO_1}, SpCl_{1_{LO_2}})$	$sim_{SpCl}(SpCl_{1_{LO_1}}, SpCl_{n_{LO_2}})$
$SpCl_{m_{LO_1}}$	$sim_{SpCl}(SpCl_{m_{LO_1}}, SpCl_{1_{LO_2}})$	$sim_{SpCl}(SpCl_{m_{LO_1}}, SpCl_{n_{LO_2}})$

If the clauses start with the same subordinate conjunction and consist only of the data domain object constructions, the similarity is computed for the DDOCs only (step 2a in Figure 4.11).

The comparison between the specifying clauses, containing both verbal and data domain object constructions is computed in the same way as for the main parts of the learning outcomes (step 2b in Figure 4.11). The only difference with respect to the main parts of the learning outcomes is the similarity measure used for comparison for the action verb constructions. In the specifying clauses we use $sim_{wStrSem}$ only to compare the action verbs. The intuition behind this is, that in most cases the specifying clauses do not contain actions, which a student must be able to perform. The verbs in the specifying clauses characterise the data domain of the learning outcome.

$$sim_{Tail} = 0.5 \times sim_{DDOC}(DDOC_{SCl_1}, DDOC_{SCl_2}) + 0.5 \times sim_{wStrSem}(AVC_{SCl_1}, AVC_{SCl_2}) \tag{4.32}$$

The similarities in Table 4.10 are computed only for the clauses, which comply with the conditions stated above. If neither of the conditions holds, similarity for the pair of clauses is set to zero (step 2c in Figure 4.11).

$$sim_{SpCl}(SCl_1, SCl_2) = \begin{cases} 0, \text{ if the conditions do not hold} \\ sim_{SpCl}(SCl_1, SCl_2) \\ \text{ if one of the conditions holds.} \end{cases} \tag{4.33}$$

Afterwards the empty rows and columns are eliminated from the Table 4.10, the numbers m and n are reduced by the number of rows and columns removed (see Equation 4.34).

Table 4.11: Comparison of the Learning Outcomes

LO_1 \ LO_2	LO_{2_1}	LO_{2_n}
LO_{1_1}	$sim_{LO}(LO_{1_1}, LO_{2_1})$	$sim_{LO}(LO_{1_1}, LO_{2_n})$
LO_{1_m}	$sim_{LO}(LO_{1_m}, LO_{2_1})$	$sim_{LO}(LO_{1_m}, LO_{2_n})$

$$\begin{aligned} m' &= m - NumNullRows \\ n' &= n - NumNullColumns \end{aligned} \tag{4.34}$$

The aggregated similarity value between all the specifying clauses of the learning outcomes is calculated using the greedy iterative strategy described in section 4.2.1 in the following way (step 3 in Figure 4.11).

$$sim_{SpCl_{aggr}}(LO_1, LO_2) = \frac{\sum Max(sim_{SpCl}(SpCl_1, SpCl_2))}{Max(m', n')} \tag{4.35}$$

4.5.2 Aggregation of the Similarities for the Learning Outcomes

In order to compute the similarity for all the learning outcomes of two educational modules or courses, we build Table 4.11, containing all of their LOs.

The total similarity $sim_{LO_{aggr}}$ for the sets of the learning outcomes LO_1 and LO_2 of educational modules or courses M_1 (C_1)and M_2 (C_2) is calculated using the maximum average strategy based on the values for the pairs of the learning outcomes.

4.6 Summary

In this chapter we described the approach to comparison of the educational courses based on ontology alignment of their modules' keywords and learning outcomes.

The assertion from Chapter 2 stated that programme specifications are too general to provide thorough information for comparison of the educational courses. However, module templates detalise the contents and students' abilities on completion of the disciplines in a more detailed way, they are closely related to the fields of studies. For this reason we decided to compare the modules, belonging to the courses, to estimate how many of them are alike or differ and to evaluate the degree of similarity.

The alignment algorithm for ontologies presented in Chapter 3 was designed. Its main aim was to compute the similarity between two educational courses or modules based on their keywords and learning outcomes.

We computed the total similarity for a pair of educational modules (courses) as an average between the keywords' and learning outcomes' similarity.

Before describing the algorithm in the detail we presented the two methods, which we utilised further on to find the average value in a matrix. They included the greedy iterative and maximum average strategies. The first of them was based on the greedy strategy. Its idea was to find all the maximum values in the matrix, crossing out their rows and columns. If the matrix was not square, the shorter side was restored until the uncrossed lines on the longer side existed. Afterwards the simple average was computed for the sum of all extracted values. The second method summed the largest values above the threshold τ in each of the rows (columns) and found the average values. The final value was computed as the simple average between the rows' and columns' average values. We also presented the algorithm for comparison, which we used to compute structural and semantic similarity between a pair of any words, independently on their part of speech.

The first stage of the algorithm was devoted to comparison of the keywords. The similarity for a pair of keywords was calculated as the simple average between their structural and semantic similarity and similarity value for all their corresponding relations (detalisation, prepositional and verbal). The similarity measure for the topics considered the similarities between all the keywords, which they contained. The final similarity between all the topics and keywords of the modules was computed in accordance with the maximum average strategy, where the threshold τ was set to 0.40.

The second stage of the algorithm compared the learning outcomes. Before describing its stages we introduced the novel similarity measure for comparison of the action verbs. We converted D. Krathwohl's taxonomy of educational aims into two-dimensional space. We based the similarity measure on the euclidean distances between the verbs in this model. We called the measure CAVe, because it reflected semantic similarity in the **c**ognitive space between the **a**ction **ve**rbs. The measure satisfied the main properties required from a similarity measure: it was symmetric, took non-negative values in the interval [0,1] and returned the value 1 for the same verbs. In order to store the coordinates of the action verbs in the two-dimensional space we created a small ontology, representing the model. This was easily integrated in the course, module and learning outcome ontology.

The learning outcomes' alignment algorithm contained the two main steps: pairwise comparison of all the learning outcomes of the educational modules (courses) and aggregation of the results using the maximum average strategy. For each pair of the outcomes the following strategy was applied. Firstly, we evaluated the similarity between the main clauses of the statements. The data domain object constructions were compared. If the similarity value was less than 0.50, the algorithm terminated returning the similarity value zero for the pair of the outcomes. In other case the similarity for the action verb constructions was calculated. The similarity for two main clauses was computed as simple average between the data domain object and action verb constructions. Afterwards the average similarity for all the specifying clauses was computed, if only both of the sentences contained the clauses, which satisfied one of the following conditions. In order to be comparable, the specifying clauses were to start either with verbs (participles) or the same specifying conjunction. The total similarity between two learning outcomes was either set to the similarity between their main clauses, or computed as a simple average between the similarities of their main and specifying clauses, where the threshold τ was set to zero.

Chapter 5

Implementation

5.1 Introduction

The main aim of this chapter is to describe how the ontologies, recognisers of the keywords and learning outcomes, the similarity measure CAVe and the ontology alignment algorithm were implemented.

First, we will present the process model of the entire system. Thereafter, we will depict ontology creation using the tool Protégé 4.0.2. Later, the applications that we created in GATE 6.1 to annotate the ”Keywords” and ”Learning Outcomes” sections of the module templates with non-terminals of the corresponding grammars will be discussed. These tags allow for the population of the ontologies. Next, the main Java packages, classes and methods, the implementation of the semantic similarity measure CAVe and the ontology alignment algorithm will be described.

5.2 The Process Model of the System

We implemented a system that allows for the comparison of the educational courses with a defined structure. This means that we treated all the modules included in the courses as if they were all compulsory for the learner, not taking into consideration whether they are compulsory, optional or belong to the elective modules in the corresponding programme specifications. The reason for this is

that the main aim of this software was to evaluate the proposed approach for the recognition of keywords and learning outcomes, as well as the algorithm for the comparison of the courses and modules based on the alignment of their ontologies. The ontologies were implemented using the Protégé 4.0.2 tool. They were accessed and processed by the algorithm with the help of OWLAPI. The ontology alignment algorithm was implemented in Java.

The process model of the system is presented in Figure 5.1. We will describe each component and transition in the system throughout this chapter. The keywords and learning outcomes recognition process and the ontology population (elements 2 and 3 in Figure 5.1) are described in section 5.3. The reasoning (element 4 in Figure 5.1) is presented in section 5.4. The "Aligner" (element 5 in Figure 5.1) is discussed in detail in section 5.5

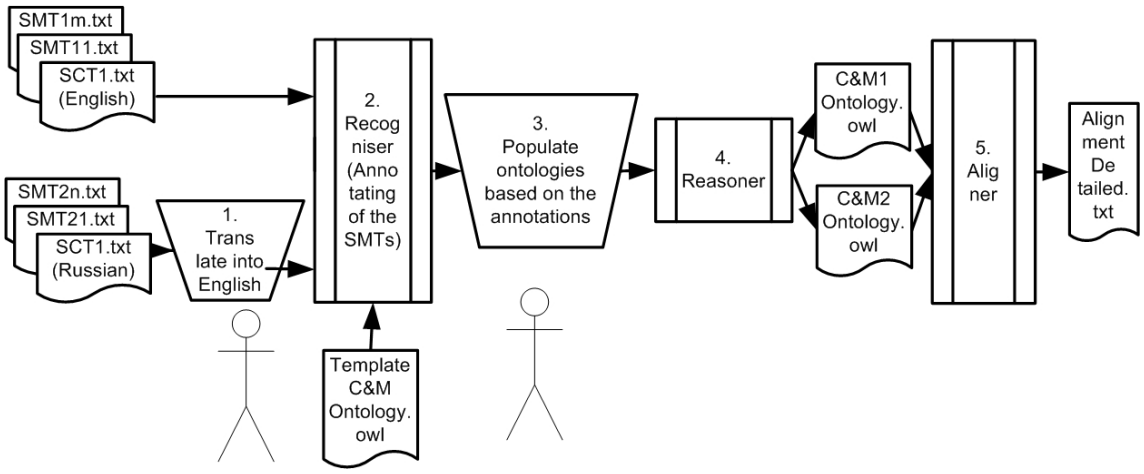

Figure 5.1: The Process Model of the Semi-automated System for Comparison of Educational Modules and Courses

As can be seen in Figure 5.1, the system receives the short course and module templates as text files in the English language as input, and an OWL Template Course and Module ontology (Template C&M ontology). The examples of short course and module templates are presented in Appendix F. Non-English documents should first be translated into the English language (step 1 in Figure 5.1).

The template C&M ontology should have the structure that was described in Chapter 3. It must include the following parts: the basic ontology of an educational course and module, including the basic data domain and learning outcomes

ontology, and the ontology of verbs, which is used to store the verbs and their coordinates in Bloom's (1956) model as revised by Krathwohl (2002).

The following classes of the template C&M ontology are to be filled with individuals before it is sent to the Recogniser for annotating of the Short Module Templates:

1. The classes that contain possible characteristics of the data domain objects: **Size**, **Importance** and **Complexity**.
2. The class **SC** containing possible subordinating conjunctions, the preposition "for" and the particle "to".
3. The classes **ClusterAV** (compulsory) **SubclusterAV** (compulsory) and **DataDomainAcionVerb** (optional) that contain possible action verbs of the learning outcomes.

In addition, the sub properties of the object properties *hasDetalisationRelation*, *hasVerbalRelation* and *hasPrepositionalRelation* may also exist.

5.3 The Recogniser and Ontology Population

In this work, the ontologies were populated manually, based on the mark-up of the natural language short course and module templates. The annotating was realised using the grammars of keywords and learning outcomes implemented in GATE 6.1.

The Recogniser in our system contains two applications, which receive the sections of the module templates that contain the keywords and the learning outcomes, and mark them with the annotations, which allow for the enrichment of the ontologies. In order to implement the Grammar of the Keywords and the Grammar of the Learning Outcomes, we used the General architecture for text engineering, which is briefly described in the following section.

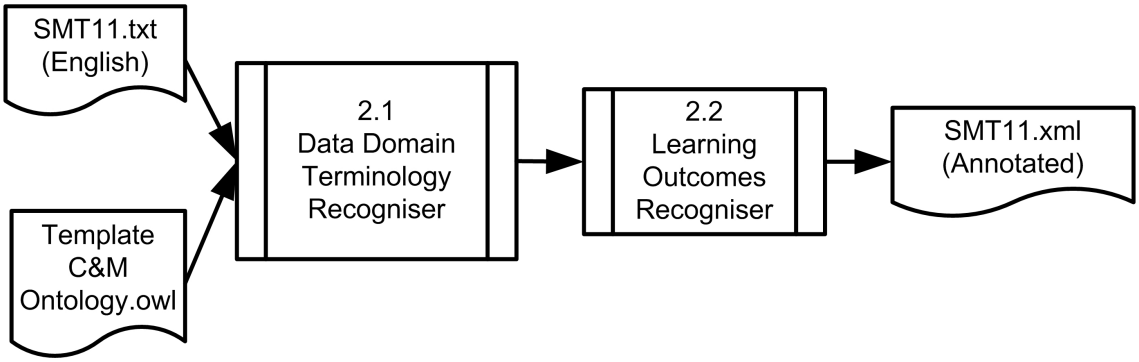

Figure 5.2: The Recogniser for a Short Module Template

5.3.1 Implementation of the Recognisers in GATE

GATE is a powerful, free open source system that allows for the development, testing and utilization of the software for processing human language. GATE consists of the two major components, namely GATE Developer and GATE Embedded. The first is a user interface that permits the addition of various existent resources for natural text processing, such as sentence splitting, part of speech recognition and the like. According to [23, 35, p.37], its basic function is the annotation of documents. The latter is an API that allows the inclusion of GATE's functionality into other applications.

In this work, we utilised GATE Developer for annotating the keywords and learning outcomes of the educational modules. As one of the targets for future work, we plan to implement the keywords' and learning outcomes' recognisers using GATE Embedded. Once combined with the alignment algorithm, programmed in Java as a part of this work, it could introduce a software tool that would receive the text documents with the LOs as its input and return the similarity values for them. In this way, the results of this research work can become of significant practical value.

GATE Developer contains the components of the following types [34].

1. Language resources (LRs), which are used to represent such resources as natural language texts, text corpora and ontologies.

2. Processing resources (PRs), which are used to perform the actions across the LRs' resources that are needed for annotating. For example, they include morphological analysers, POS-taggers and so on.

3. Visual resources (VRs) are used for the visualisation and editing of the components in GUI. For example, it is possible to edit an ontology by adding or modifying its entities.

According to [35, p.72], "the set of resources integrated with GATE is known as CREOLE: a Collection of REusable Objects for Language Engineering".

A GATE application is a sequence of processing resources that can be applied to texts in order to annotate them.

GATE provides several types of applications, which differ in their approaches to the loading and processing of documents. They include the following types.

1. **Pipeline** identifies applications that can be run across one document only.
2. **Corpus Pipeline** identifies applications that can be run across a set of documents.
3. **Conditional Pipeline** is used when there is a need to run not all of the processing resources based on the document's features.
4. **Conditional Corpus Pipeline** is the same as the previous example, but can be run across a set of documents.

We will not stop at depicting the wide variety of GATE's processing resources that can be used for manipulation with the natural language texts. Instead, we will describe the applications that we created for the annotation of these keywords and learning outcomes and will briefly describe only the processing resources that they include.

Both of the recognisers contain processing resources of the following types.

Document Reset This resource removes all the annotations from the texts.

ANNIE Sentence Splitter A sentence splitter is used to distinguish sentences in a text. It uses a set of transducers and gazetteers for this purpose. As used here, ANNIE stands for "a Nearly-New Information Extraction System" [34].

ANNIE English Tokeniser The tokeniser is used to annotate words, numbers, symbols, spaces and punctuation marks. For example, it can differentiate words with initial letters in upper or lowercase, which may be helpful for further application of the grammar's rules.

ANNIE POS Tagger is used to annotate texts that contain parts of speech.

GATE Morphological Analyser takes as input a document annotated with parts of speech. It identifies the lemma, affix and root. Thereafter, this information can be used for further annotation, such as the comparison of words based on their roots.

Gazetteer allows the identification of entities in the text that are from other sources, including plain lists of words and ontologies. For instance, the ANNIE Gazetteer contains a predefined set of lists, including currencies, dates and so on. Onto Root and Flexible Gazetteers are subsequently used to annotate texts with ontology entities.

JAPE Transducer is a processing resource that allows the execution of the JAPE grammars, where JAPE stands for Java Annotation Patterns Engine. According to [35, p.190], "JAPE allows you to recognise regular expressions in annotations on documents".

A JAPE grammar consists of several phases, each of which may contain several rules. A rule, in turn, has left and right hand sides (LHS and RHS, respectively). The LHS contains an annotation pattern description, while the RHS consists of manipulation statements. The grammar phases run sequentially and annotate the texts. Each phase has three compulsory parameters that need to be set. The first is the title of the phase, which is used as its identifier. The second is called "Input", and is used to define the titles of the annotations to be used in the rules. The third parameter is "Control", which can have one of the following values: all, brill, first, once or appelt. The "all" control style applies all the rules to the matching parts of the text. This means that one sequence of words (or symbols) may be annotated several times using one or different rules. The "brill" style also applies all the matching rules, but advances matching from the position at which the longest match finishes. The "first" style means that a rule fires only for the first found match and does not try to find a longer match. The "once" style terminates the annotation process when it finds the first match for the rule. The

"appelt" style means that only one rule can be applied to one region of text in accordance with the priorities of the rules. Prioritisation works in the following way. If several rules can be applied to the regions of the text, starting from the same point, the one that will produce the longer match is fired. In the event that the regions are of equal length, the rule with highest priority is applied. If several rules with the same priority can be used to annotate the same piece of text, the one defined earlier in the grammar is used.

It can be seen that a recogniser based on the JAPE-grammars could encounter several variants of ambiguity when processing a statement. Two main approaches are used to solve this problem. The first is grouping the rules into phases in such a way that firing each subsequent set uses the results of the previous transduction. This enables the addition of a precondition to the rule, in that a word or a sequence of words can only be annotated with a non-terminal of the grammar if it was or was not annotated with a different non-terminal during the preceding phases. The second approach includes assigning priorities to the grammar rules (in this case, the "appelt" control style is chosen). This means that, in the event of ambiguity, the rule with the higher priority is fired.

5.3.2 Recognition of the Keywords

We produced an application of the type "Corpus Pipeline" for the annotation of the sections "Keywords" and "Learning Outcomes" of a short module template. We chose this type of application because we needed to run it across a set of documents while using the same grammatical rules for each. We first looked for keywords in both of the meaningful sections of the SMT, in accordance with the rules of the Grammar of a Keyword.

The application consists of the following processing resources: Document Reset, ANNIE Sentence Splitter, ANNIE English Tokeniser, ANNIE POS Tagger, GATE Morphological Analyser and six JAPE Transducers, which sequentially apply the rules of the Grammar of a Keyword. We utilised the processing resources that already exist in GATE 6.1 and implemented the rules of our Grammar in the JAPE language. The phases of the grammar's implementation in JAPE are presented in Appendix G.

5.3.3 Recognition of the Learning Outcomes

We produced an application of the type "Corpus Pipeline" for the annotation of the learning outcomes because we needed to run our application across a set of documents.

The application consists of the following processing resources: Document Reset, ANNIE Sentence Splitter, ANNIE English Tokeniser, ANNIE POS Tagger, GATE Morphological Analyser, Flexible Gazetteer and JAPE Transducers, which implement the rules of the Grammar of the Learning Outcomes. We utilised the processing resources that already exist in GATE 6.1 and implemented the rules of our Grammar in the JAPE language.

The use of the first five processing resources is standard, while the latter two require some detailed explanation.

The application contains the language resources with the ontologies of the modules. At this stage, the ontology must satisfy the following requirements:

1. The ontology must include the topics, keywords and relationships based on the annotations produced by the recogniser of the keywords.
2. The ontology classes, the identifying characteristics of a data domain object (Importance, Complexity and Size), contain the individuals.
3. The class **SC** must contain the instances of the words that can begin a specifying clause.
4. The ontology must contain the adjusted Krathwohl model, including at least the basic verbs used in the model.

A pair of Onto Root and Flexible Gazetteers was created for each of the modules' ontologies. They allow for the annotation of the learning outcomes with the ontology's entities, which are used for the further mark-up process.

The rules should be fired in a particular order, because each subsequent group uses the results of its predecessor. For this reason, the application of the rules of the Grammar of the Learning Outcomes was divided into several phases. The numbering of the rules mentioned in the phase's description is in accordance with the Grammar of the Learning Outcomess specification in Section 3.4.5. We created the package of phases in JAPE grammar, which allows for the annotation of the learning outcomes. These are presented in Appendix G.

5.3.4 Population of the Common Classes of the C&M Ontology

The user should create individuals of the class **Programme Specification** and may input the information regarding it in accordance with the SCT.

The expert should create an individual of the class **Course** as the course title and link it to the corresponding PS. The data properties ”Course Code”, ”CCredit”, ”Faculty”, ”Department” and ”University” should be completed from the corresponding lists. The relations ”hasCourseLevel” and ”hasCProfile” should be assigned. The titles of the modules of the course must be added as individuals of the class **Module**. The data properties ”Module Code”, ”MTitle” and ”MCredit” are set. The relations *hasMProfile* and *hasMLevel* are assigned.

Should the Learning Outcomes Recogniser find action verbs and present participles that are not present in the class **ActionVerb**, these must be added to the ontology as individuals of the class **DataDomainActionVerb**. An expert must assign them to the instances of the class **SubclusterAV** through the object property *isDDAVInSubcluster*. The values of the properties *DDAVCPD* and *DDAVComplexity*, identifying their coordinates in the adjusted Krathwohl model of educational aims, are inferred by the reasoner based on these relations.

5.3.5 Population of the Data Domain Sub-ontology

The data domain part of the C&M ontology is populated based on the annotations produced by the Keywords recogniser in the following sequence:

1. The phrase annotated as ”Topic” becomes an individual of the class ”Topic”, as does ”Keyword”, because as class Topic is its subclass.
2. All non-duplicate phrases annotated ”Keyword” become individuals of the class **Keyword**. Single abbreviations are added as Keywords. If an abbreviation phrase is detected, the full title is added to the class **Keyword**, while the abbreviation is input as the value of its data property *Abbreviation*.
3. All non-duplicated words annotated ”AdjDD” and ”NounDD” are added as instances of the classes with the same names.

4. All non-duplicating words marked as "PrepRel", "DetRel" and "VerbRel" are added as sub properties of the relations *hasPrepositionalRelation*, *hasDetalisationRelation* and *hasVerbalRelation*, respectively.

5. The object properties *hasPrepositionalRelation*, *hasDetalisationRelation* and *hasVerbalRelation* are assigned in order to link the individuals of the class **Keyword** in the same way as that in which they are annotated in the text.

6. The object properties *containsAdjDD* and *containsNounDD* are set between the individuals of the class **Keyword** and classes **AdjectiveDD** and **NounDD**, respectively, based on the annotations.

7. The property *hasKeyword* is set between an individual of the class **Topic** and all the Keywords that appear between the colon (after the topic's title in the text) and the nearest full stop.

5.3.6 Population of the Learning Outcomes Sub-ontology

The learning outcomes part of the C&M ontology is populated based on the annotations produced by the recogniser in the following sequence:

1. The number of the learning outcomes is counted. For each, an identifier (ID) is created and is added as an individual of the class **Learning Outcome**. The pattern is the following *LOi*, where *i* is the number of the module's learning outcomes. The relation *hasLearningOutcome(Module, LO_i)* is assigned.

   For each of the learning outcomes:

   (a) The property *hasActionVerb (LO_i, AV_{i_j})* is assigned based on the recognised action verbs of the learning outcome. Here, AV_{i_j} is an individual of the class **ActionVerb**.

   (b) An ID of a data domain object construction is created as an individual of the class **DDOConstruction**. The pattern is *DDOCi*, where *i* is the number of the module's learning outcomes. The property *hasDDOConstruction (LO_i, $DDOC_i$)* is assigned.

   For each data domain object construction:

i. The number of the data domain objects is calculated. An ID is created for each. The pattern is *DDOij*, where *i* is the number of the module's learning outcomes and *j* is the number of the data domain objects. The object property value *hasDataDomainObject (DDOC$_i$, DDO$_{i_j}$)* is added.
For each data domain object:

A. If a characteristic of the data domain object was detected during the annotation, the object property *hasComplexity (DDO$_{i_j}$, Complexity$_i$)*, *hasImportance (DDO$_{i_j}$, Importance$_i$)* andor *hasSize (DDO$_{i_j}$, Size$_i$)* is (are) assigned.

B. The relation *hasDDO (DDO$_{i_j}$, Keyword)* is created between the instances of the data domain objects and the individuals of the class **Keyword** of which the corresponding DDOs consist.

(c) the number of the specifying clauses is calculated. For each of the specifying clauses of the learning outcomes, an SClause ID is created as an instance of the class **SClause**. The pattern is the following: *SCl$_{i_j}$*, where *i* is the number of the module's learning outcomes and *j* is the number of its specifying clauses. The relation *hasSClause (LO$_i$, SCl$_{i_j}$)* is set.

For each of the specifying clauses:

i. If the SClause begins with a subordinating conjunction *SC$_{i_j}$*, then the relation *hasSC (SClause, SC$_{i_j}$)* is added to the ontology.

ii. Should the specifying clause begin with a sequence of action verbs or present participles, the relations *hasSCActionVerb (SClause$_{i_j}$, AV$_{i_{j_k}}$)* are set. Here *AV$_{i_{j_k}}$* stands for an individual of the class **ActionVerb**.

iii. The identifier for the data domain object construction of the specifying clause is added as an instance of the class **DDOConstruction**. The pattern for the name is *SCDDOCij*, where *i* is the number of the learning outcomes and *j* is the number of its specifying clauses. The data domain objects contained in this construction are added in the same way as for the main clause. The pattern for ID is *SCDDOijk*, where *i* is the number of the learning outcomes and *j* is the number of its specifying clauses. *k* is the number of the objects inside the construction.

5.4 Reasoning

We utilised the reasoner Pellet in the system. The template C&M Ontology contains the following SWRL-rules, which allow automatic inference of certain information. The rules declare the following.

1. The values of the properties *DDAVCPD* and *DDAVComplexity* that identify coordinates in the adjusted Krathwohl model of the educational aims are inferred by the reasoner, based on the relations *isDDAVinSublcuster*.
2. If a course contains a module and the module has certain learning outcomes, then the relation *hasCourseOutcome (Course, LearningOutcome)* is set.

The reasoner also infers the following, based on the ontology's structure:

1. Adds all the individuals of the classes **Topic**, **AdjectiveDD** and **NounDD** to the class **Keyword**, based on the class hierarchy;
2. Adds relation *hasRelation* between all the individuals linked through the properties *hasPrepositionalRelation*, *hasDetalisationRelation* and *hasVerbalRelation*, based on the hierarchy of properties;
3. Adds relation *belongToTopic* between all the Keywords and Topics, based on the values of the relation *hasKeyword*.

5.5 Implementation of the Ontology Alignment Algorithm

The system for comparison of the educational modules and courses, using the algorithm presented in the Chapter 4, is implemented in Java. The ontologies are stored as .owl files on disc and are accessed using the methods of OWLAPI. The alignment part of the system is presented in Figure 5.3.

We used the following, existing free libraries in the implementation.

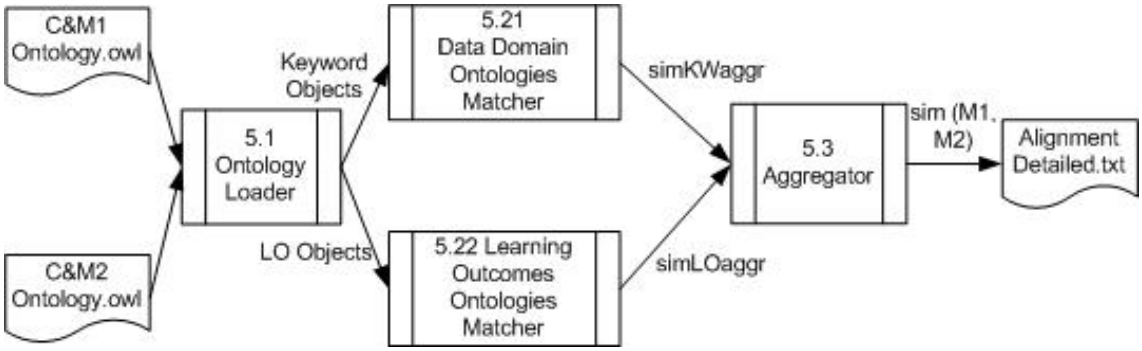

Figure 5.3: The Aligner for a Pair of Short Module (Course) Templates

1. **OWLAPI** was used to process the ontology.
2. **Pellet** was utilised to reason across the ontology [104].
3. **Simmetrics** library was used as a source of Levenshtein edit distance and Soundex + Jaro-Winkler similarity measures.
4. **Ontosim** was used for learning whether two words were synonyms in WordNet. Its implementation of the Wu and Palmer similarity measure was also utilised.
5. **DISCO** library was used as a source of the DISCO2 similarity measure.

The application contains four packages:

1. **DMU.BMSTU.CC.Components** contains the classes that reflect the main components of the ontology. It is used to download the input ontologies.
2. **DMU.BMSTU.CC.Similarity** implements the ontology alignment algorithm.
3. **DMU.BMSTU.CC.Util** contains the classes and methods that apply minor transformations to the data used by the core components of the system.
4. **DMU.BMSTU.CC.Test** contains the tests for the experiments.

5.5.1 The Loader

The "Ontology Loader" is used to create Java objects from the ontology individuals. We utilised it as a set of Java classes in a separate package, "DMU.BMSTU.CC.Components". The package contains classes, each of which contains the constructor for presenting an ontology entity and its properties in the manner required by the alignment algorithm.

The package "DMU.BMSTU.CC.Components" contains the following classes.

Keyword.java

DataDomainObject.java

ActionVerb.java

SClause.java

LearningOutcome.java

EduModule.java

EduCourse.java

In order to load an educational module or course, we need to load its keywords and learning outcomes. This is done in the following way.

The Loader of a Keyword

For each keyword, we created an object of the Java class Keyword, as presented in Figure 5.4 and described below.

First, we entered the following variables (steps 1-4 in Figure 5.4).

- *KeywordStr* is the string representation of the ontology instance of the ontology class **Keyword**.
- *AdjDD* is the string representation of the ontology instance of the ontology class **AdjDD** if the keyword contains an adjective.

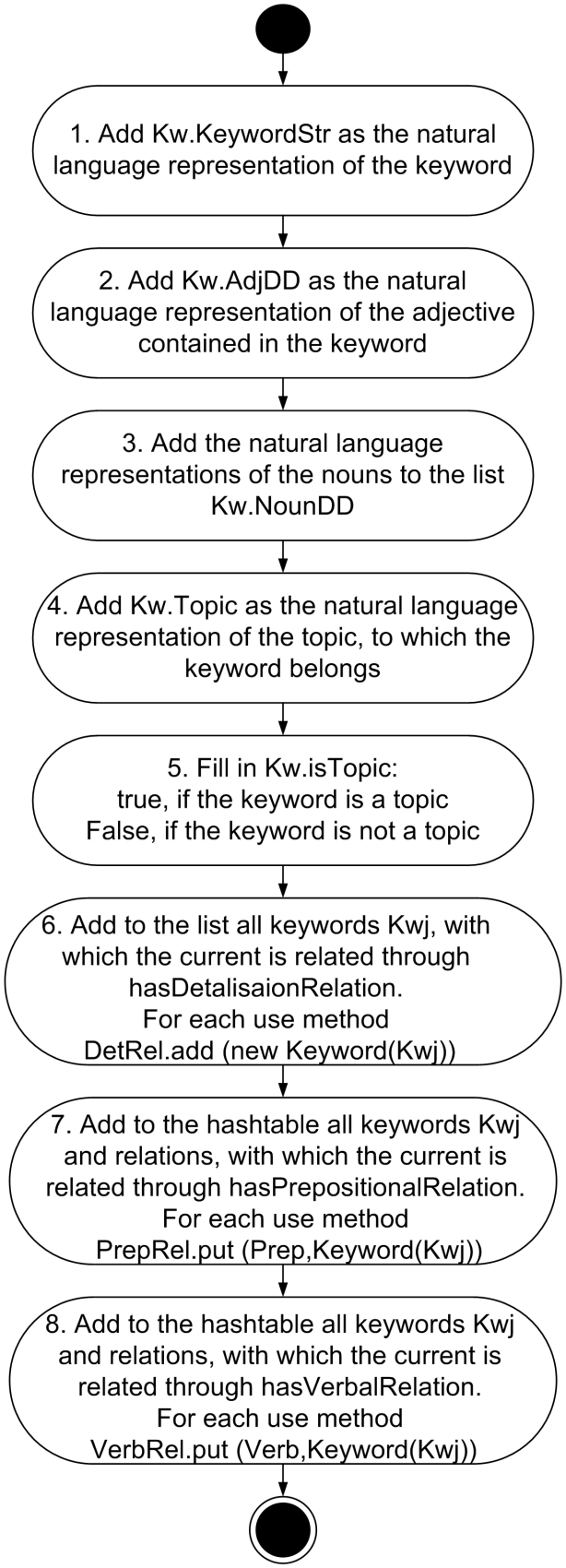

Figure 5.4: The Loader of a Keyword

- *LinkedList < String > NounDD* is the list of the string representations of ontology instances of the ontology class **NounDD**.
- *Topic* is the string representation of the title of the topic to which the keyword belongs.

We set the Boolean variable as *isTopic* true if the keyword belonged to the ontology class **Topic**, and false if it did not (step 5 in Figure 5.4).

We then extracted the relations between the current keyword and the others. For the *hasDetalisationRelation*, we formed a list of the keywords to which the initial one is related (step 6 in Figure 5.4). For the *hasPrepositionalRelation*, we stored both the preposition and the target keyword (step 7 in Figure 5.4). For the hasVerbalRelation, we also preserved the verb and the targeted keyword (step 8 in Figure 5.4).

The Loader of a Learning Outcome

For each learning outcome, we created an object of the Java class LearningOutcome, as presented in Figure 5.5 and described below.

Firstly, we entered the list of the action verbs using objects of the class ActionVerb (step 1 in Figure 5.5). We included the following variables for each object:

- *ActionVerbStr* is the string representation of the ontology instance of the ontology class **ActionVerb**.
- *CPD* is the coordinate in the cognitive process dimension of the action verb.
- *Complexity* is the complexity of the action verb.

Secondly, we loaded all the data domain objects that belong to the data domain object construction of the learning outcome (step 2 in Figure 5.5). For each object, we entered the following variables:

- *LinkedList < Keyword > KW* is the list of the keywords that the data domain object contains.

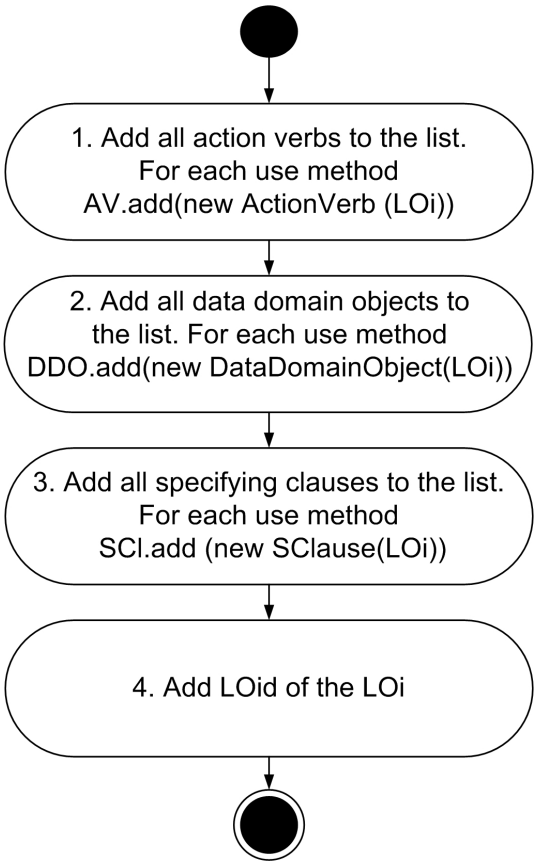

Figure 5.5: The Loader of a Learning Outcome

- *Size* is the size characteristic of the data domain object in the ontology.
- *Complexity* is the complexity characteristic of the data domain object in the ontology.
- *Importance* is the importance characteristic of the data domain object in the ontology.

Thirdly, we loaded the specifying clauses of the learning outcomes, if any (step 3 in Figure 5.5). In order to do this we created an object of the class SClause. For each object, we entered the following variables:

- *SC* is the natural language representation of the subordinate conjunction (individual of the ontology class **SC**).
- *LinkedList< ActionVerb > SCActionVerbs* is the list of the action verbs of the specifying clause.
- *LinkedList < DataDomainObject > SCDDO* is the list of the data domain objects of the specifying clause.

Finally, we added the LOid to the learning outcome (step 4 in Figure 5.5).

5.5.2 Data Domain Sub-ontologies Matcher

This matcher is used to compare the keywords of the educational modules or courses. In order to compare the sets of keywords of two educational modules or courses, we considered those that are recognised as topics and keywords in the "Keywords" sections of the short module templates.

The algorithm for the calculation of the similarity between the pair of keywords is utilised twice in the system. Firstly, it is used for the pairs of keywords and, secondly, for the data domain objects of the learning outcomes. The computation is realised through the classes and methods of the package "DMU.BMSTU.CC.Similarity". Each pair of keywords contains the following stages. Steps 1-4 are illustrated in Figure H.1, while step 5 is presented in the Figure H.2.

1. If both keywords contain adjectives (”AdjDD”), the structural and semantic similarity *simAdjDD* between them is computed using the method *KeywordSimilarity. CalculateAdjectiveToAdjectiveSimilarity*.

2. If both keywords contain nouns (”NounDD”), the structural and semantic similarity between them is computed. For each pair of nouns, the similarity is computed using the method *KeywordSimilarity. CalculateNounToNounSimilarity*. The matrix containing these values is then filled in. The average value is calculated in accordance with the greedy iterative algorithm using the method *ResultMatrix. CalculateGreedyIterative*.

3. The weights for the adjectives and nouns contained in the keywords are calculated in accordance with the algorithm presented in Section 4.3.1.

4. The $sim_{Kw_{StrSem}}(Kw_1, Kw_2)$ is computed in accordance with the Formula 4.6.

5. The similarity between the relations of the keywords is computed.

6. The similarity between all the keywords of the topics, to which the keywords belong is calculated by the method *CalculateTopicToTopicByKeywordsSimilarity*.

7. The total similarity of the pair of keywords is calculated in accordance with the Formula 4.20.

The similarity of all the keywords of the two educational modules (courses) is computed in accordance with the algorithm presented in Figure 4.4.

5.5.3 Learning Outcomes’ Sub-ontologies Matcher

This matcher implements the CAVe similarity measure for the action verbs and for all the intermediate stages for the computation and aggregation of the similarities between the learning outcomes and their constituents.

We created a separate class, KrathwohlSimilarity.java, which receives the ontology that contains the adjusted Krathwohl taxonomy’s model. It computes the CAVe similarity between the action verbs.

The realised algorithm for the comparison of a pair of learning outcomes is presented in Figure H.3. It implements the algorithm presented in Figure 4.10.

Firstly, the main clauses of the learning outcomes are compared and the specifying clauses follow, but only if they comply with the conditions presented in the section 4.5.

The similarities between the data domain object constructions are calculated in accordance with the algorithm presented in the section 4.5. The algorithm presented in the section 5.5.2 is utilised for the pairs of the keywords that are contained in the data domain objects. The implementation thereof was presented in the previous Section.

The similarities of the action verb constructions are computed according to the algorithm presented in section 4.5. The CAVe similarities for the pairs of single action verbs are computed dynamically, based on the pre-computed model of the verbs.

The total similarity of a pair of learning outcomes is calculated based on the similarity values of their main and specifying clauses, and is calculated in accordance with the algorithm presented in section 4.5.

The aggregated similarity of all the learning outcomes of the modules is computed in accordance with the maximum average strategy. Only those pairs of outcomes in which data domain objects constructions' similarities exceed the value 0.50 are considered.

5.5.4 The Aggregator and the Output

The final aggregation of the value computes the total similarity for a pair of modules/courses as the average value between the similarities of their keywords and learning outcomes.

The algorithm produces a text file on the run. It contains the total similarity values for all the keywords, learning outcomes and overall modules. In addition, the document contains all the detailed information regarding the comparison. For example, it is possible to see not only the final similarity, but also all the constituents of each pair of the keywords, including the structural and semantic similarity values and the similarities between each of the relations. With regard to the learning outcomes, it is also possible to see all the intermediate similarity values between all the data domain object and action verb constructions, thus getting to deeper levels of their single objects and verbs.

5.6 Summary

In this chapter, we explained how the ontologies and grammars that were presented in Chapter 3 were implemented in the systems Protege 4.0.2 and GATE 6.1, respectively. We also described the realisation, in Java, of the alignment algorithm from Chapter 4.

The template ontologies of a course, a module and a learning outcome, a field of study and of verbs were created once in Protégé 4.0.2. In later stages, they were reused by being populated from different documents.

The system takes as input the basic ontologies, short course and module templates, which should contain general information such as the title, the level and the credits. The module templates should also have sections with keywords and learning outcomes. The basic data, as well as the course structure, will be entered manually into the ontology. The sections ”Keywords” and ”Learning Outcomes” should be sent to the corresponding Recognisers realised as GATE applications. We utilised standard GATE components to build these applications. We implemented the rules of the Keywords' and Learning Outcomes' grammars using JAPE rules. The annotated sections of texts should be manually entered into the learning outcome and field of study parts of ontology. Thereafter, the reasoner is run to execute SWRL-rules and to infer additional knowledge.

The similarity measure CAVe and the alignment algorithm were implemented in the Java language, using OWLAPI to process and Pellet to reason the ontology. The libraries Simmetrics, Ontosim and DISCO were utilised because they implemented similarity measures. The measure CAVe was identified as a separate class. It contains the methods to compute maximum distance in the model, and the distance and similarity CAVe for a pair of action verbs. Its methods should receive ontology using Krathwohl's taxonomy as input, as presented in Chapter 4. The ontology should contain the action verbs to be compared. For a pair of educational modules (courses), the algorithm returns a .txt file, which contains their total similarity and a highly detailed report regarding similarities between each pair of keywords, learning outcomes and their constituents.

Chapter 6

Evaluation

6.1 Introduction

This chapter is devoted to the evaluation of the results of this research. Our first goal is to assess the quality of the annotation of the keywords and learning outcomes by the corresponding recognisers. The second aim is to evaluate the algorithm for the comparison of educational courses and modules, based on ontology alignment.

We evaluated the annotation and alignment results based on the module templates provided by the De Montfort and the Bauman Moscow State Technical Universities.

For the first assessment, we processed the ”Keywords” and ”Learning Outcomes” sections of the Module Templates using the GATE applications, as described in Chapter 5. We computed such quantities as Recall, Precision and F-measure using the AnnotationDiff tool provided by GATE 6.1.

We evaluated the alignment algorithm by comparing its results to those of human judgement. We provided pairs of single learning outcomes, as well as five pairs of short module templates containing keywords and learning outcomes. The comparison is performed by groups of experts and by the algorithm. Following this, we evaluated the results.

6.2 Description of the Data Sets

The evaluation of the systems is divided into two sections. Firstly, we evaluated the quality of the annotation of the keywords and learning outcomes in the corresponding sections of the modules templates. Secondly, we evaluated the algorithm's performance compared to human judgement of the modules' similarities.

In order to evaluate the recogniser and the algorithm, we produced three groups of documents.

The first group contains a set of five pairs of statements that contain one or more learning outcomes. The second and third groups contain the module templates from the De Montfort (DMU) and the Bauman Moscow State Technical Universities (BMSTU). The BMSTU templates were translated from Russian into English for these experiments. We divided them into two groups according to their data domain. In each of the sets, we assigned one "master" module to which all the others were compared. The first set contains four modules on data models and databases, while the second includes three modules devoted to object-oriented programming in Java. We converted all the module templates to shorter versions that contained only the keywords and the learning outcomes.

6.3 Evaluation of the Recogniser

In order to evaluate the quality of the recognition, we used the AnnotationDiff tool that is provided by GATE for such purposes. We compared the same documents, one of which was annotated by hand ("gold standard"), while the other was annotated by the application.

In order to describe the quality of the annotation, we present the number of the entities annotated, the number of correctly extracted items, the number of partially correct mark-ups (when the annotation intersects with the desired piece of text), missing and false positives (when a wrong piece of text is annotated) in the tables below. The AnnotationDiff tool also produced the values for Recall, Precision and the F-measure, which are the quantities used to depict the quality of the results. We used "strict" methodology to compute all the values, which means

that partially correct entities are treated as mistakes. Below, we will explain the equations that the tool utilised to compute the corresponding quantities. These were presented in the GATE User Guide ([36, p. 245]).

The first measure, Recall, depicts the completeness of the set of annotated sections of text. It reflects the ratio of correct tags to the total number of sections of text that were to be annotated. This measure does not take into account the number of false positive tags that were assigned to the word sequences. Recall is calculated in accordance with Formula 6.1, where $|Correct|$ stands for the number of correct annotations, $|Missing|$ shows how many tags were lost by the recognizer and $|Partial|$ reflects the number of partially correct annotations.

$$Recall = \frac{|Correct|}{|Correct| + |Missing| + |Partial|} \tag{6.1}$$

Precision is the second measure used to evaluate the accuracy of the annotators. It is calculated as the ratio of correct tags to the total number of annotations produced by the recogniser (see Formula 6.2). The concepts $|Correct|$ and $|Partial|$ are the same as in Equation 6.1. The constituent $|FalsePositive|$ represents the number of tags that were wrongly assigned to the sections of text.

$$Precision = \frac{|Correct|}{|Correct| + |FalsePositive| + |Partial|} \tag{6.2}$$

The F-measure is used to balance the first two quantities by computing the weighted average of Precision and Recall (see Formula 6.3). The coefficient β^2 reflects the weighting of Precision versus Recall in Equation 6.3. In our work, we set the value of β to 1 after having assigned equal weights to both characteristics.

$$F - measure = \frac{(\beta^2 + 1) \times Precision \times Recall}{(\beta^2 \times Precision) + Recall} \tag{6.3}$$

6.3.1 Evaluation of the Annotations for the Keywords

Table 6.1 represents the results for the topics, keywords and their constituents. The row "Number" represents the number of annotations of a certain type (for example, there are 34 annotations labelled "Topic"). The row labelled "Correct"

Table 6.1: Evaluation of the Annotations (Keywords)

Parameter	Topic	Keyword	AbbrPhr	NounDD	AdjDD
Number	34	325	7	495	43
Correct	32	314	7	495	43
Partially correct	0	9	0	0	0
Missing	0	3	0	10	9
False Positive	2	2	0	0	0
Recall	1	0.96	1	0.98	0.83
Precision	0.94	0.97	1	1	1
F-measure	0.97	0.96	1	0.99	0.91

shows the number of annotations that were correctly assigned. The row labelled ”Partially correct” reflects the number of partially correct annotations. ”Missing” indicates the number of annotations that were lost by the recogniser, while ”False Positive” indicates the number of wrongly annotated words/word combinations. The value in the cell ”Number” equals the sum of ”Correct”, ”Partially correct” and ”False Positive” annotations.

As can be seen in Table 6.1, the precision, recall and f-measure are high for all categories. The performance of the annotator was worst for the AdjDDs (adjectives), where the recall is 0.83. However, having checked the original texts, we noticed that in almost all of the cases the adjective ”relational” was being lost due to the wrong POS-tag.

Table 6.2 illustrates the results for different types of relationships. It can be seen that the number of extracted and actual detalisation and verbal relations is relatively small. The quality of the annotation was good for detalisation and prepositional relation phrases, as supported by the high values of precision, recall and the f-measure. At the same time, the recogniser performed worse with verbal relation phrases, which was mainly caused by incorrect POS-tagging.

6.3.2 Evaluation of the Annotations for the Learning Outcomes

Table 6.3 illustrates the situation regarding the learning outcomes and their main and specifying clauses. The quantities show that the quality of recognition is very good in all cases and is excellent for the specifying clauses.

Table 6.2: Evaluation of the Annotations (Relations)

Parameter	Detalisation Relation Phrase	Prepositional Relation Phrase	Verbal Relation Phrase
Number	2	47	7
Correct	2	47	4
Partially correct	0	0	0
Missing	0	2	1
False Positive	0	0	3
Recall	1	0.96	0.80
Precision	1	1	0.57
F-measure	1	0.98	0.67

Table 6.3: Evaluation of the Annotations (Learning Outcomes)

Parameter	Learning Outcome	Main Clause	Specifying Clause
Number	57	58	40
Correct	56	57	40
Partially correct	1	0	0
Missing	0	0	0
False Positive	1	1	0
Recall	0.98	1	1
Precision	0.97	0.98	1
F-measure	0.97	0.99	1

Table 6.4: Evaluation of the Annotations (Learning Outcomes' Constituents)

Parameter	Action Verb	Action Verb	Data Domain Construction	Data Domain	$Keyword_{LO}$
Number	70	94	113	152	201
Correct	68	91	112	151	198
Partially correct	0	0	1	1	2
Missing	0	0	0	0	0
False Positive	2	3	1	0	3
Recall	1	1	0.99	0.99	0.99
Precision	0.97	0.97	0.98	0.99	0.98
F-measure	0.99	0.98	0.99	0.99	0.98

Table 6.4 shows the results for the constituents of the main and specifying clauses of the learning outcomes. All the quantities are very close to 1, which indicates a very high quality of annotation.

6.4 Evaluation of the Alignment Algorithm

6.4.1 Evaluation Methodology

The selected evaluation methodology was a comparison of the results produced by the algorithm to those of human judgement. This approach was used due to the nature of the research task, which included semantic similarity computation. The other reason for choosing this method of evaluation is that the algorithm is designed to assist human experts to make decisions.

We designed a questionnaire that contains the same questions for each pair of modules to be compared (see Appendix I). We asked the experts to evaluate whether they thought two modules were similar in terms of their keywords and learning outcomes, as well as if they were generally similar. The experts were provided with a Likert scale [18, 24, 60, 83] that contained five options, namely "Strongly disagree", "Disagree", "Neutral", "Agree" and "Strongly agree". Each expert was then asked about the level of confidence in his or her answer ("Low", "Moderate" or "High"). Only those answers that were scored "Moderate" or "High" in terms of confidence were included in further computations.

Table 6.5: Assignment of the Intervals of the Algorithm's Similarity Values to the Answers on the Likert Scale

Likert Scale	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Coordinate	1	2	3	4	5
Interval	0 - 0.19	0.20 - 0.39	0.40 - 0.59	0.60 - 0.79	0.80 - 1

Alignment of the Scales

The intervals of similarity values produced by the ontology alignment algorithm were assigned to the possible answers on the Likert scale that was completed by the experts.

In Table 6.5, we assigned coordinates and intervals to each of the possible answers across the Likert scale. The coordinates were later used to compute the descriptive statistics of the experts' replies. The intervals were used to interpret the values returned by the alignment algorithm. As can be seen in Table 6.5, the results of the human answers and the computer calculation are matched in such a way that each answer has a corresponding interval of values that may be produced by the algorithm. This alignment is also useful for the interpretation of the results produced by the algorithm by the supposed user of a system that can utilise it.

The Experts' Choices

In order to evaluate the results produced by the algorithm, we used module templates on databases, data modelling and object-oriented programming from undergraduate courses. We wished to obtain the opinions of people that we consider to be prospective users of our system, such as academic staff, PhD students, business people who may wish to study further and undergraduate students. For this reason, we approached people from the above listed categories who were expert in the field of information technology. We were interested in receiving evaluation results from the international community. Thus, we obtained replies from people in the United Kingdom, Europe and the Russian Federation. We also wanted to examine the ways in which the replies of more experienced experts differed from those of less experienced people. Therefore, we differentiated between two groups of experts.

The first group consisted of 24 people and represented academic staff and PhD-students from the United Kingdom and Europe and who specialise in computer science, and people working in the IT industry in Russia. The British and European respondents received the questionnaire and materials in English, while the Russians were supplied with a version in Russian. 61% of the experts from the first focus group admitted that they had experience in either the teaching or supervision of students, or in the design of educational modules, or both. The second group included 38 IT undergraduate students from Russia who were studying the modules that they were asked to compare. Both groups of experts contained male and female representatives.

Mann-Whitney U Test

Because we had two groups of experts, we wanted to determine whether their opinions regarding the similarity of keywords, learning outcomes and modules followed the same tendency in each of the experiments. If this were the case, we would be able to combine their answers and analyse them in one group. In order to determine whether this assumption held true, we needed to select a suitable statistical test to formulate the null and alternative hypotheses and to see if we could reject the first in favour of the second at the certain confidence level [37]. The hypotheses are as follows.

The Null Hypothesis

The distribution of the answers across the Likert scale is alike for the both groups of experts.

The Alternative Hypothesis

The distribution of the answers across the Likert scale differs for the two groups of experts.

The null hypothesis states that both groups of experts would evaluate the similarity between the keywords, learning outcomes and the complete educational module in approximately the same way. In other words, the same percentage of respondents from each of the groups would select the same options each of the five possible answers. The alternative hypothesis asserts the opposite.

The Mann-Whitney U Test (also known as the Mann-Whitney Wilcoxon, or MWW test) is a non-parametric test of the null hypothesis that two samples come from the same population. If the null hypothesis is rejected at a chosen confidence level, the alternative hypothesis is considered to be true. The alternative hypothesis states that a particular sample has greater values than does the other. However, in our case we need to conduct a simple test, because we only want to see if the replies from both groups of experts follow the same tendency. The MWW test is suitable when the variables are not normally distributed. This is exactly the situation here, because it is difficult to predict how human replies will be distributed across the Likert scale. The distribution may differ from one question to another.

The MWW test entails the following main assumptions. Our data satisfy all of them.

Assumption 1 The dependent variable should be measured at the ordinal or continuous level. The Likert scale satisfies this.

Assumption 2 The independent variable should contain independent categorical groups. We had two independent groups of experts.

Assumption 3 The observations should be independent of each other. In our research, each of the experts belonged to only one of the groups of experts.

The Mann-Whitney U test consists of the following steps.

We need to first state the null and alternative hypotheses. In terms of the MWW test, our hypotheses can be reformulated in the following way.

The Null Hypothesis (MWW)

There is no difference between the ranks of the answers across the Likert scale for both groups of experts.

The Alternative Hypothesis (MWW)

There is difference between the ranks of the answers across the Likert scale for the two groups of experts.

Secondly, we must decide whether a one- or two-tailed test is to be used. We also must choose the confidence level for the confidence interval and the corresponding significance level α. The significance level is the complement of the respective confidence level. We state our confidence interval at the 0.95 confidence level,

as is typically done in statistical analysis. In this case, the significance level α equals 0.05. We prefer the two-tailed test to the one-tailed test, because we are performing a non-directional test. We are attempting to see if both samples could come from the same population and not to determine which particular sample has larger values than does the other.

Thirdly, we must choose one of the two possible variants of the U-statistics' computation. Each will be chosen based on the sample size. For samples smaller or equal to 21, we could use the U-tables to see if the null hypothesis could be rejected. However, in each of the experiments, we collected a larger number of responses in each group. There are no values in the U-tables for such cases. Therefore, as the U-statistics are approximately normally distributed, we must compute the z-value and consult the z-table. For the confidence level of 0.95, we can reject the null hypothesis if the z-value is less than -1.96 or if it exceeds the value 1.96. Otherwise, the null hypothesis cannot be rejected at this confidence level.

The algorithm for computing the U-statistics and the z- and corresponding p-values is the following. We will provide a small walk-through example. Although each of the example's samples consists of three elements only, it is applicable to evaluate them using the z-ratio. Small samples can be analysed either by using the U-statistics and the U-table, or by computing the z-ratio and consulting the Z-table.

1. The data from samples A and B should first be grouped into one ordered set, set C. For example, if sample A contained $n_A = 3$ three elements (3;4;5) and sample B contained $n_B = 3$ three elements (3;4;4), then set C would consist of six elements, (3;3;4;4;4;5).

2. The data in set C should be ranked from the smallest to the largest. In the event of tied ranks (when there are several identical values of the replies), the average rank is assigned to each of them. For example, set C will be ranked as follows: the first number is the value taken from the set and the second is its rank. Thus, (3 - 1.5; 3 - 1.5; 4 - 4; 4 - 4; 4 - 4; 5 - 6).

3. The data are then returned to their original sets. In our example, set A will be (3 - 1.5; 4 - 4; 5 -6) and set B will be (3 - 1.5; 4 - 4; 4 - 4). The sums of the ranks R_A and R_B for each of the samples are computed. In our example,

$R_A = 11.5$, $R_B = 9.5$. The assumption is that, if the samples are alike, the sums of their ranks should not differ significantly.

The mean rank for the two samples A and B is computed in accordance with the following Equation 6.4.

$$R_{AB} = \frac{n_A + n_B + 1}{2} \tag{6.4}$$

If the null hypothesis were true and the two samples did not differ significantly, we would expect the average rank for each of them to approximate the overall mean value R_{AB}. For this reason, R_{AB} is used to compute the expected sums of ranks μ_{R_A} $\mu_{R_B}$for the samples A and B in the following way.

$$\mu_{R_A} = n_A \times R_{AB} \mu_{R_B} = n_B \times R_{AB}$$

In our example, $R_{AB} = 3.5$. Thus, $\mu_{R_A} = \mu_{R_B} = 3 \times 3.5 = 10.5$.

4. The next step is to calculate the standard deviation, which will be the same for each of the sampling distributions R_A and R_B. It is computed in accordance with Equation 6.5.

$$\sigma_R = \sqrt{\frac{n_A \times n_B \times (n_A + n_B + 1)}{12}} \tag{6.5}$$

In our example, $\sigma_R = \sqrt{\frac{3 \times 3 \times (3+3+1)}{12}} = 1.87$.

5. The penultimate step is to compute the z-ratio for both samples in accordance with Equation 6.6. We must add 0.5 to the numerator if $R_i < \mu_{R_i}$, and decrease the numerator by 0.5 if $R_i > \mu_{R_i}$. In accordance with [77], this is done so as to correct for continuity, because the sampling distributions are intrinsically discreet.

$$z_A = \frac{(R_A - \mu_{R_A}) \pm 0,5}{\sigma_R} z_B = \frac{(R_B - \mu_{R_B}) \pm 0,5}{\sigma_R} \tag{6.6}$$

In our example $z_A = \frac{(11.5-10.5)-0.5}{1.87} = 0.27$, $z_B = \frac{(9.5-10.5)+0.5}{1.87} = -0.27$.

Table 6.6: Comparison of the Pairs of Learning Outcomes

Number of Pair	# of responses	Algorithm	Median	Mode	Range
1	38	4	4	4	3
2	38	3	2	2	4
3	38	4	3	4	4
4	38	4	3	4	4
5	38	3	3	4	3

6. The final step is the interpretation of the results. The table p-value for z_A and z_B for a two-tailed test equals 0.78, which is greater than 0.05. Moreover, the z-values -0.27 and 0.27 lie between the critical z-values -1.96 and 1.96. This means that, in our example, we cannot reject the null hypothesis because there is no difference between the ranks of the answers across the Likert scale for the groups of experts; thus, both samples have similar distributions.

When discussing the MWW test later in this work, we will provide the z-ratio and the p (2)-value for the two-tailed test.

6.4.2 Experiment using Pairs of Learning Outcomes

Statisticians agree that there is no point in calculating the mean, the standard deviation and the variance of data collected using the Likert scale. It is advisable to analyse the median, the mode and the range instead.

This experiment was only evaluated by the second group of the experts, the Russian undergraduate students. The descriptive statistics are presented in Table 6.6. Each pair of learning outcomes was evaluated by 38 respondents. The column "Algorithm" represents the value returned by the algorithm. In order to intepret the double value produced by the programme as one of the answers across the Likert scale, we used Table 6.5. The columns 4-6 in Table 6.6 present the median (the middle value in the sample), the mode (the most frequently met value) and the range (the difference between the largest and the smallest values in the sample) of the samples. All the characteristics were computed after the conversion of the experts' replies to the Likert coordinates based on Table 6.5.

Table 6.7: Comparison of the Pairs of Learning Outcomes

Number of Pair	Index	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
1	# % Alg	0 0%	7 18%	9 24%	18 47% × (0.68)	4 11%
2	# % Alg	3 8%	21 55% × (0.36)	6 16%	7 18%	1 3%
3	# % Alg	2 5%	12 32%	6 16% × (0.44)	13 34%	5 13%
4	# % Alg	1 3%	11 29%	8 21%	17 45% × (0.66)	1 3%
5	# % Alg	0 0%	5 13%	16 42%	16 42% × (0.63)	1 3%

In Table 6.7 and in the remainder chapter, the symbol # stands for the number of the respondents' answers in each category. The symbol % denotes the percentage of the answers in each category, assuming that the total number of answers is 100%. The abbreviation "Alg" is used for the value produced by the algorithm. The symbol × is entered into the cell in which the result falls. The numeric value is given in brackets.

In order to discuss the results, it is necessary to look more closely at the learning outcomes that were compared.

The First Comparison

The first pair consists of the following learning outcomes.

$NLLOSentence_1$: Implement SQL scripts for data retrieval and data modification.

$NLLOSentence_2$: Create SQL scripts for database table creation, database table modification, data insertion and data modification.

Each of the sentences contains only one learning outcome, which contains main and comparable specifying clauses. The value 0.68 returned by the algorithm falls

into the interval "Agree", and thus coincides with the mode and the median of the experts' answers.

The Second Comparison

The second pair of learning outcomes is represented by the following sentences.

$NLLOSentence_1$: Discuss Relational Database Management Systems, such as Oracle.

$NLLOSentence_2$: Create and check a database structure.

Each of the sentences contains one learning outcome. The first LO consists of the main and specifying clauses, while the second LO contains only the main clause. For this reason, the specifying clause of the first outcome is ignored by the algorithm. The result, 0.36, falls into the "Disagree" interval and coincides with the most popular reply, as well as with the mode and median of the experts' opinions.

The Third Comparison

The third comparison contains the following sentences.

$NLLOSentence_1$: Produce moderately complex data models.

$NLLOSentence_2$: Understand and discuss the data models of databases and the data models of knowledge bases.

Both of the sentences contain a single learning outcome, consisting of the main clause only. The result of 0.44 produced by the algorithm falls into the interval "Neutral". The mode of the results given by the experts equals 4 ("Agree"), while the median is 3 ("Neutral"). It can be seen that 13 people chose the answer "Agree", while as many as 12 replied "Disagree". This means that the experts did not come to a joint decision. This result could possibly be the result of different approaches in their judgement.

In order to analyse the algorithm's behaviour, we should look more closely at its intermediate results. On one hand, both outcomes have similar data domain objects, containing the common denominator "data models". On the other hand, the actions that a student should be able to apply differ significantly. The algorithm returned the similarity value 0.66 for the data domain object constructions, which can be interpreted as "Agree[ing]" that they are alike. At the same time, the

similarity between the action verb constructions equals 0.22, which means that they "Disagree" on their similarity. This means that the intermediate stages of the algorithm returned reasonable results. The high similarity value for the data domain object constructions, combined with the low similarity value for the action verb constructions, brings the total result to 0.44, which we interpret as falling into the "Neutral" interval. Considering the low level of agreement among the experts, we evaluated the result of the comparison of this pair of learning outcomes as acceptable.

The Fourth Comparison

The fourth pair contained the following sentences.

$NLLOSentence_1$: Produce a database structure and create the database structure in a Relational Database Management System, such as Oracle.

$NLLOSentence_2$: Create a database structure in a Relational Database Management System, such as Access.

It can be seen that the first sentence contains two learning outcomes. The first consists of the main clause only, while the second contains a specifying clause as well. In this example, we evaluate how well the algorithm responds to such cases. The value of 0.66 produced by the programme falls in the interval "Agree" and coincides with the mode of the poll's results.

The Fifth Comparison

The fifth comparison consists of the following sentences.

$NLLOSentence_1$: Apply UML for object-oriented models creation and use Java Software Development Kit Application Programming Interfaces.

$NLLOSentence_2$: Create UML models and implement, check and document Java programmes.

In this case, each of the sentences contains two learning outcomes. Here, we evaluate how well the system deals with the comparison of small sets of learning outcomes. The algorithm produced the value 0.63, which appears close to the lower border of the "Agree" interval. As can be seen from the table, an equal number of experts (16) chose the answers "Agree" and "Neutral" where the mode identifies the interval "Agree" and the median "Neutral". The result of the

Table 6.8: Comparison of the Module Templates. Experiment 1

Parameter	Expert Group	Median	Mode	Range
Keywords	AcB	3	2	4
	RS	3	4	2
	Both	3	4	3
Learning Outcomes	AcB	4	4	4
	RS	3.5	4	3
	Both	4	4	4
Overall	AcB	4	4	3
	RS	3	3	3
	Both	3	4	3

algorithm coincides with the mode and follows its tendency to be closer to the interval ”Neutral”.

Thus, the comparison of the experts' evaluation with the values computed by the alignment algorithm permit us to claim that it is capable of producing reliable results. This position is strengthened by the fact that we examined the algorithm's performance in different scenarios, such as when the learning outcomes contained main clauses only or when they also contained specifying clauses, when the sentences being compared contained one or two learning outcomes each or when they had one LO in the first and two LOs in the second sentence, and when the experts replied ”Agree”, ”Disagree” or did not reach a consensus.

6.4.3 Experiments with the Modules

Experiment 1

The first pair contains the modules ”Data Analysis and Database Design” and ”Database Design Concepts”. The short modules can be found in Appendix I.

It can be seen that the groups of experts expressed different opinions regarding the similarity of the keywords. The first group (academic staff, PhD students and business representatives) mainly disagreed that they could be judged as being similar. However, it is important to note that the differences in the number of answers in the categories ”Disagree”, ”Neutral” and ”Agree” is somewhat vague:

Table 6.9: Comparison of the Module Templates. Experiment 1

Parameter	Expert Group	Index	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
Keywords	AcB	#	1	8	5	7	3
		%	4%	33%	21%	29%	13%
	RS	#	0	10	8	16	0
		%	0%	29%	24%	47%	0%
	Both	#	1	18	13	23	3
		%	2%	31%	22%	40%	5%
	Alg					× (0.63)	
Learning Outcomes	AcB	#	1	4	5	12	2
		%	4%	17%	21%	50%	8%
	RS	#	0	12	5	14	3
		%	0%	35%	15%	41%	9%
	Both	#	1	16	10	26	5
		%	2%	28%	17%	45%	9%
	Alg					× (0.60)	
Overall	AcB	#	0	5	3	13	3
		%	0%	21%	13%	54%	13%
	RS	#	0	7	14	11	2
		%	0%	21%	41%	32%	6%
	Both	#	0	12	17	24	5
		%	0%	21%	29%	41%	9%
	Alg					× (0.62)	

29% for ”Agree”, 21% stayed ”Neutral” and 33% ”Disagreed”. At the same time, the undergraduate students chose the answer ”Agree” in most cases (47 %). As a result of these circumstances, the total value fell into the interval ”Agree”. However, the MWW-test returned the following results for the ”Keywords”: the z-value equals 0.21 and p (2) = 0.8337. As the z-ratio lies between the critical values -1.96 and 1.96 and the p-value exceeds 0.05, we cannot reject the null hypothesis, which states that the two samples are alike. The difference in the distribution in the two groups is insignificant; thus, we can analyse the combined replies of both groups. The similarity produced by the algorithm equals 0.63, which is interpreted as ”Agree” and coincides with the experts' joint opinion.

As can be seen in Tables 6.8 and 6.9, both groups of experts came to the same conclusion in that they ”Agree” that the learning outcomes of the modules are similar. The algorithm produced the value 0.60, which falls into the interval ”Agree” and which also coincides with the mode of the answers (4, ”Agree”). The MWW test's z-value for the ”Learning Outcomes” equals -0.65 and p (2) = 0.5157. The z-ratio falls between the critical z-values and p (2) exceeds 0.05. Thus, we cannot reject the null hypothesis that the two samples are alike. The difference in distribution in the two groups is insignificant; thus, they can be analysed together.

The overall result for this pair of modules is ”Agree” for the professionals and ”Neutral” for the undergraduates. The Mann-Whitney U test returned the z-value -1.49 and p (2) = 0.1362. The z-ratio lies between the critical values and $p > 0.05$. For this reason, we cannot reject the null hypothesis that the two samples are alike. The different distribution in the two groups is insignificant; thus, we can analyse the results together. 41% of the summed numbers of replies fell into the interval ”Agree”. The algorithm returned the value 0.62, which is interpreted as ”Agree”.

Experiment 2

The second experiment compared the modules ”Data Analysis and Database Design” and ”Data Models”.

The results of the second comparison, illustrated by the data in Tables 6.10 and 6.11, show that the two groups of experts had different opinions regarding the

Table 6.10: Comparison of the Module Templates. Experiment 2

Parameter	Expert Group	Median	Mode	Range
Keywords	AcB	3	2	3
	RS	3	4	4
	Both	3	4	4
Learning Outcomes	AcB	2	2	4
	RS	3	2	4
	Both	3	2	4
Overall	AcB	3	2	3
	RS	3	4	3
	Both	3	4	3

Table 6.11: Comparison of the Module Templates. Experiment 2

Parameter	Expert Group	Index	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
Keywords	AcB	#	2	8	6	7	0
		%	9%	35%	26%	30%	0%
	RS	#	1	7	9	13	2
		%	3%	22%	28%	41%	6%
	Both	#	3	15	15	20	2
		%	5%	27%	27%	36%	4%
		Alg				× (0.63)	
Learning Outcomes	AcB	#	3	10	5	4	1
		%	9%	35%	26%	30%	0%
	RS	#	1	13	7	10	1
		%	3%	41%	22%	31%	3%
	Both	#	4	23	12	14	2
		%	7%	42%	22%	25%	4%
		Alg		× (0.37)			
Overall	AcB	#	1	10	6	6	0
		%	4%	43%	26%	26%	0%
	RS	#	1	8	11	12	0
		%	3%	25%	34%	38%	0%
	Both	#	2	18	17	18	0
		%	4%	33%	31%	33%	0%
		Alg			× (0.51)		

similarity of the ”Keywords” sections. However, it can be seen that the representatives of academia and those of the IT industry vacillated regarding their replies. The difference between the number of answers in the categories ”Disagree”, ”Neutral” and ”Agree” is just one vote. At the same time, the undergraduate students showed more confidence in their answers and gave 41% of their votes to the category ”Agree”. The MWW-test resulted in a z-value of 1.55 and p (2) = 0.1211. Again, the z-value fell between its critical values and the p-value exceeded the significance level of 0.05. We could not reject the null hypothesis that the two samples were alike. The distributions in the two groups differed insignificantly. In total, 36% of all the respondents agreed that the keywords of these modules were similar, which produced the mode 4 (”Agree”). The algorithm performed in the same way, having returned the similarity value 0.63, which falls into the ”Agree” interval.

Both groups of experts showed more conformity regarding the learning outcomes of the educational modules. 35% of the respondents from the AcB and 41% of undergraduates from the corresponding groups evaluated them as being dissimilar, selecting the answer ”Disagree”. The z-value for the ”Keywords” equals 1.16 and p (2) = 0.246. Thus, we cannot reject the null hypothesis that the two samples are alike. The distributions in the two groups differ insignificantly. The algorithm returned the value 0.37, which is accepted as falling into the interval ”Disagree” and thus coincides with the experts’ opinion.

The overall impression of the modules’ similarity differed across the two groups of respondents. 43% of the academic and business people voted for ”Disagree”, while 38% of the students inclined to the answer ”Agree”. However, the results of the MWW test showed that the distributions in the two groups differed insignificantly (z-value equals 1.26 and p (2) = 0.2077). Thus, we cannot reject the null hypothesis that the two populations are alike. The summed results of the votes show that the experts did not come to a joint opinion regarding this question, as an equal number of replies (33 %) fell into the intervals ”Disagree” and ”Agree”. The algorithm returned the value 0.51, which was interpreted as ”Neutral”. Due to the uncertainty of the poll’s results, we judge the algorithm’s behaviour applicable in such cases, because it correctly advises to the user to make a decision on his or her own.

Table 6.12: Comparison of the Module Templates. Experiment 3

Parameter	Expert Group	Median	Mode	Range
Keywords	AcB	4	4	4
	RS	3	4	4
	Both	4	4	4
Learning Outcomes	AcB	3	4	4
	RS	3	3	3
	Both	3	4	4
Overall	AcB	3	4	4
	RS	3	4	4
	Both	3	4	4

Table 6.13: Comparison of the Module Templates. Experiment 3

Parameter	Expert Group	Index	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
Keywords	AcB	#	1	6	4	12	1
		%	4%	25%	17%	50%	4%
	RS	#	1	7	8	12	3
		%	3%	23%	26%	39%	10%
	Both	#	2	13	12	24	4
		%	4%	24%	22%	44%	7%
		Alg				× (0.60)	
Learning Outcomes	AcB	#	2	6	6	8	2
		%	8%	25%	25%	33%	8%
	RS	#	1	9	11	10	0
		%	3%	29%	35%	32%	0%
	Both	#	3	15	17	18	2
		%	5%	27%	31%	33%	4%
		Alg			× (0.43)		
Overall	AcB	#	1	6	6	10	1
		%	4%	25%	25%	42%	4%
	RS	#	1	9	9	11	1
		%	3%	29%	29%	35%	3%
	Both	#	2	15	15	21	2
		%	4%	27%	27%	38%	4%
		Alg			× (0.52)		

Table 6.14: Comparison of the Module Templates. Experiment 4

Parameter	Median	Mode	Range
Keywords	2	2	3
Learning Outcomes	2	2	4
Overall	2	2	4

Experiment 3

In this example, both groups of respondents (50% and 39% for the AcB and the undergraduates, respectively) chose the answer ”Agree” regarding the similarity of the modules' keywords. The MWW test proved that the distributions in the two groups differed insignificantly (the z-value equals 0.04 and p (2) = 0.9681). The high p-value indicates that we cannot reject the null hypothesis that the two samples are alike. The algorithm produced the value 0.60, which is interpreted as ”Agree”, and thus coincides with the respondents' opinion.

The results of comparison of the learning outcomes differed slightly in both groups. 35% of the students voted ”Neutral”, and almost the same number fell into the categories ”Agree” and ”Disagree” categories (32% and 29%, respectively). At the same time, 33% of the academic staff and IT business representatives decided to ”Agree”, giving 25% of their votes to each of the categories ”Disagree” and ”Neutral” each. The MWW test showed the following results: the z-value equals -0.42 and p (2) = 0.6745, which give us grounds for claiming that the two samples are alike. This means that neither group of experts could confidently decide if they agreed or disagreed regarding the similarity of the learning outcomes. The sum of the numbers of the replies resulted in 33% of the votes for the ”Agree”, 31% for the ”Neutral” and 27% for the ”Disagree” categories. The algorithm returned the value 0.43, which is treated as ”Neutral”. Once again, we find it applicable in such cases of uncertainty.

Experiment 4

The results of the poll showed that the experts chose the answer ”Disagree” in each question of this experiment. 52% disagreed regarding the similarity of the keywords and 57% disagreed on the similarity between the learning outcomes.

Table 6.15: Comparison of the Module Templates. Experiment 4

Parameter	Index	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
Keywords	# % Alg	1 4%	12 52%	4 17% × (0.58)	6 26%	0 0%
Learning Outcomes	# % Alg	4 17% ×(0.07)	13 57%	3 13%	2 9%	1 4%
Overall	# % Alg	1 4%	13 57% × (0.33)	4 17%	4 17%	1 4%

Table 6.16: Comparison of the Module Templates. Experiment 5

Parameter	Median	Mode	Range
Keywords	2	2	3
Learning Outcomes	2	2	3
Overall	3	3	3

Consequently, 57% of respondents disagreed on the overall similarity. The algorithm produced the value 0.58 for the keywords, which falls into the category ”Neutral” and which is a little too high in this case, as it does not reject the similarity of the ”Keywords” sections of the module templates. The algorithm for the alignment of the learning outcomes returned the value 0.07, which falls into the category ”Strongly disagree”. This value does not coincide with the mode of the poll’s results. However, it confidently rejects the similarity between the outcomes of the modules. We consider the algorithm’s behaviour applicable in such a case, as the respondents’ opinion was also significantly inclined towards disagreement (74% of the total answers fell into the categories ”Disagree” and ”Strongly disagree” on the total). The overall similarity value for the two modules equals 0.33, which falls into the category ”Disagree”, which coincides with the respondents’ most popular opinion.

Table 6.17: Comparison of the Module Templates. Experiment 5

Parameter	Index	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
Keywords	# % Alg	3 14%	8 38%	5 24% × (0.54)	5 24%	0 0%
Learning Outcomes	# % Alg	4 19%	8 38% × (0.36)	5 24%	4 19%	0 0%
Overall	# % Alg	3 14%	7 33%	7 33% × (0.45)	4 19%	0 0%

Experiment 5

In the last experiment, 38% of the respondents chose the category ”Disagree” to characterise the dissimilarity between the keywords of the modules. At the same time, an equal number of answers (24% of the total of each) were assigned to the variants ”Agree” and ”Neutral”. The algorithm returned the value 0.54, considered to be ”Neutral”, which may be applicable in this case as the difference between the numbers of answers in the categories is rather small. With regard to the learning outcomes, 38% of the respondents chose the answer ”Disagree”, while 24% stayed ”Neutral”. However, in this case, 57% of the total fell into the categories ”Disagree” and ”Strongly disagree”. The algorithm resulted in 0.36, which means ”Disagree” in this case and which coincides with the experts’ opinion. Comparing the overall impression, the respondents gave an equal number of votes for the categories ”Neutral” and ”Disagree” (33% for each). The algorithm returned the value 0.45 (”Neutral”), and thus coincided with the experts’ opinion.

6.5 Critical Discussion of the Results

The evaluation of both the Keywords’ and the Learning Outcomes’ annotators, as well as of the ontology alignment algorithm, proved that both produced reasonable results and can thus be utilised by information systems that may require such functionality. We provide a detailed discussion and critique of the results, below.

6.5.1 Critical Discussion of the Annotators' Results

The evaluation of the Keywords' and Learning Outcomes' annotators showed the following.

The topics, keywords and their constituents were recognised with a high level of precision: 32 topics out of 34, 314 keywords out of 325 and all abbreviation phrases, nouns and adjectives were recognised correctly. At the same time, the recall of the adjectives was somewhat low and equalled 0.83, with nine of the tags missing. The analysis showed that, in seven cases, the adjective "relational" was not annotated because the POS-tagger identified it as a noun. In other cases, the mistakes were also due to the wrong POS-tags. Erroneous tagging of the adjectives led to annotations of the keywords that were partially correct.

The detalisation and prepositional relation phrases were recognised correctly (precision equalled 1). This was because the constituents of these relations, namely keywords, prepositions and exact phrases like "such as" and "for example", were easy to annotate. At the same time, the annotator for verbal relation phrases performed seriously worse. The precision thereof equalled 0.57, while the recall was 0.80. The present or past participles were recognised as verbs in all the false positive phrases, while in reality they were being used as nouns; for example, the 'UML modelling technique'. Once again, incorrect POS-tags resulted in incorrect annotations.

Recognition of the learning outcomes performed well. Recall varied from 0.97 to 1, while the precision fell into the interval from 0.98 to 1. Only one of the 57 learning outcomes and its main clause were recognised partially correctly. All the specifying clauses were annotated without mistakes. None of the learning outcomes' constituents was missing. However, two partially correct $Keyword_{LO}s$ led to one partially correct data domain object and data domain object construction. Three false positive action verb annotations were found. These problems were interconnected. The mistakes were again due to the wrong POS-tags, because past participles were treated as verbs instead of as adjectives and a noun was mistaken for a verb. However, these errors did not significantly affect the overall result.

In summary, the annotators for the Keywords and Learning Outcomes performed well, having shown a high level of precision, recall and the F-measure. In order to raise the quality of recognition, it is recommended that the POS-tagger be improved.

6.5.2 Critical Discussion of the Algorithm's Results

We paid much attention to the evaluation of the ontology alignment algorithm. The difficulty lay in the fact that what we had designed was not a common algorithm that could be applied to any ontology and could thus be evaluated by the standard procedures recommended by the Ontology Alignment Evaluation Initiative. Moreover, and to the best of our knowledge, we could not find any analogous systems that could perform a comparison of educational courses, modules and learning outcomes. Due to the nature of the research, we decided to evaluate the results against those produced by human judgement. The data sets and the choice of experts were discussed in the above sections.

We first evaluated the algorithm for the matching of the individual learning outcomes. We prepared five pairs of outcomes that differed in their structure and semantics, and thus in the approach to alignment and comparison.

The first pair was used to test the alignment of two outcomes with main and comparable specifying clauses. 47% of the experts agreed that they were similar. The algorithm produced the value 0.68, which fell into the same interval.

The second pair tested the comparison of the outcomes when one had a specifying clause and the other did not. Thus, only the main sections were aligned. However, this did not affect the quality of the performance. 55% disagreed that the statements were similar. The same result was returned by our system (0.36, or disagreement).

The third pair contained two outcomes, consisting of the main clauses only with a complicated action verb and data domain object constructions. The algorithm stayed neutral regarding their similarity, having returned the value 0.44. Most of the experts' replies (34%) fell into the interval "Agree". At the same time, it is important to note that almost the same number of replies (32%) was allocated to the variant "Disagree". Consequently, we claim that the algorithm performed well, because it reflected the uncertainty of the respondents.

The fourth pair of sentences tested how well the algorithm dealt with the comparison of complicated outcomes. The first sentence contained two LOs, of which the second consisted of the main and specifying clauses. The second statement contained one outcome, containing the main and subordinate clauses. 45% of the

respondents agreed regarding their similarity. The algorithm obtained the same result, with a value of 0.66.

The fifth pair aimed at the comparison of statements that consisted of two learning outcomes. In this way, it tested not only pairwise alignment, but also the aggregation of similarities for the LOs. An equal number of respondents chose the answers ”Neutral” and ”Agree”. The algorithm returned the value 0.63, which falls into the ”Agree” interval, but which is also very close to the upper boundary of ”Neutral”.

On balance, the algorithm performed well, as in most cases its results coincided with the replies of the experts. We also consider it to be an advantage that the algorithm tends to return results in the neutral interval when none of the replies receives significantly more votes than do others.

We then evaluated the entire algorithm based on a comparison of educational modules. Five experiments were conducted. The first three pairs of the modules were evaluated by two groups of experts: Russian undergraduate students and British academic staff, international PhD students and representatives of the IT industry. Although the replies from both groups differed in several cases, they generally followed the same tendency. The Mann-Whitney U Test showed that the samples of answers of both groups had either similar or identical distributions; thus, the difference was insignificant. Therefore, we combined the replies and analysed them together.

In all three experiments, most of the experts (40%, 36% and 44%, respectively) ”Agree[d]” that the ”Keywords” sections of the templates were similar.

In the first experiment, 45% of the respondents ”Agree[d]” that the learning outcomes were similar. The algorithm confirmed this result. In the second experiment, 42% of the experts disagreed regarding the similarity of the LOs. The algorithm again returned a value belonging to the same ”Disagree” interval. In the third experiment, 33% of the experts ”Agree[d]” regarding the similarity of the learning outcomes. The algorithm returned the value 0.43, which is within the ”Neutral” interval. However, it can be seen that 5% of the experts replied ”Strongly Disagree” and 27% ”Disagree”, which means that a total of 32% of them were more inclined towards disagreement. At the same time, 33% replied ”Agree” and 4% ”Strongly agree”, bringing the total value of those who tended

to agree up to 37%. We assumed that the 5% difference between the number of respondents who chose diametrically opposed replies shows that there was no consistency their answers. This means that this particular instance requires a more detailed analysis by the experts in the field. By returning the value ”Neutral”, the algorithm demonstrated this need.

The overall impression of the similarity of the modules was as follows. In the first experiment, both the experts and the algorithm agreed that the module templates were similar. In the second case, an equal number of experts chose the answers ”Agree” and ”Disagree” (33% each). The algorithm returned the value 0.51, interpreted as ”Neutral”. We find this result acceptable because it represents the disagreement in the respondents’ opinions.

The second group of modules was evaluated by the second group of experts only (British academic staff, international PhD students and representatives of the IT industry). The results were as follows.

52% and 38% of respondents chose the answer ”Disagree” to the question regarding the keywords’ similarity in the fourth and fifth experiments, respectively. However, the algorithm returned the values 0.58 and 0.54 (”Neutral”), which gained 17% and 24% of the votes in each case. We must admit that the results of the poll and those produced by the system differed significantly.

57% of the experts considered the learning outcomes of the compared modules to be dissimilar by replying ”Disagree” in the fourth experiment. The algorithm resulted in the value 0.07, which is interpreted as ”Strongly Disagree”. We judge this outcome to be acceptable, because it reflects the fact that the outcomes are dissimilar. In the fifth experiment, the results produced by the algorithm fell within the same interval as most of the replies of the respondents (38%).

The overall estimation of the modules’ similarity by the experts was reproduced by the system. In the fourth experiment, the respondents chose the answer ”Disagree”. In the fifth experiment, an equal number of replies were given to the variants ”Neutral” and ”Disagree”. The algorithm returned the values 0.33 (”Disagree”) and 0.45 (”Neutral”) in both tests.

In summary, we can conclude the following.

The algorithm for the comparison of the learning outcomes performed well and proved to be applicable. It reflected the experts’ agreement and disagreement

regarding the similarity of the learning outcomes. The experiments conducted using the pairs of individual outcomes and with the sets of LOs proved that the algorithm coped well with both. Moreover, the system returned a "Neutral" value in cases in which the respondents could not come to a consensus. We see this as an advantage, because our algorithm outlines the cases in which a greater amount of attention from the specialist is needed.

The algorithm for the comparison of the "Keywords" section tends to return more positive results than negative ones. It performed well in the cases in which the experts agreed or stayed neutral, but was not efficient in experiments in which the experts disagreed that the modules' keywords were similar. This means that the algorithm for the comparison of the keywords needs to be improved. On the other hand, the learning outcomes' alignment utilises the same method for the comparison of the individual data domain objects. This leads us to the conclusion that we need to improve the manner in which we aggregate the similarity values, but not the algorithm for the computation of the similarities between the pairs of the keywords.

In four experiments out of five, the overall similarity between modules, calculated by our system, fell into the same interval as the experts' answers. It is interesting to note that also in four out of five cases, the respondents' estimation of the overall modules' similarity fell into the same interval as that of their keywords and learning outcomes. This result gave us grounds to assume that the keywords and learning outcomes equally affected their opinion regarding the modules' similarity. Thus, we were correct in assigning equal importance to both criteria in our algorithm.

6.6 Summary

The main aim of this chapter was to evaluate how well the proposed approach performed. The results were assessed based on the module templates provided by the De Montfort and the Bauman Moscow State Technical Universities. We created seven short module templates containing the sections "Keywords" and "Learning Outcomes". We also chose five pairs of individual learning outcomes to test how well the algorithm would perform when analysing them. The templates were divided into two groups in accordance with their data domains. The first

set contained four modules on data models and databases, while the second included three modules on object-oriented programming in Java. Each of the groups contained one ”master” template to which all the others were compared.

The evaluation consisted of the two main parts. We first checked the quality of the tags provided by the Keywords and Learning Outcomes annotators. Secondly, we compared the results produced by the ontology alignment algorithm against that produced by human judgement concerning the similarity of the pairs of modules.

The recognisers were evaluated using the AnnotationDiff tool, built in the GATE 6.1. system. We prepared the ”gold standard” annotated versions of all seven module templates and an additional ten individual outcomes. Following this, the Annotators ran and recognised the text. We evaluated the quality of tagging for each of the parameters that were to be input into the ontology. The AnnotationDiff returned the numbers of correct, partially correct, missing and false positive tags. It also computed the precision, the recall and the F-measure for each case. The results proved the applicability of the annotators, as the precision and recall values attained 0.96 to 1 for the keywords and 0.97 to 1 for the learning outcomes. The worst results were obtained for *AdjDD* (recall 0.83) and *Verbal relation phrases* (recall 0.80 and precision 0.57) in the ”Keywords” sections. Both problems were entirely due to tagging the incorrect part of speech. Adjectives were tagged as nouns and participles. Therefore, the annotator could not apply the appropriate rules and could not identify them correctly.

We approximated the evaluation of the alignment algorithm to the comparison of the pairs of educational modules, treating them as small courses, each of which contained one discipline. This approach was chosen because we had to evaluate the results against human judgement. We provided the experts with the short module templates and a questionnaire. The respondents were to express their opinion regarding the similarity between the modules using the Likert scale. We asked the experts if they ”Strongly disagree[d]”, ”Agree[d]”, stayed ”Neutral”, ”Agree[d]” or ”Strongly agree[d]”, that two modules were similar. We treated the scale as interval data, whereby all the replies covered equal intervals of possible values returned by the algorithm (”Strongly disagree”: 0-0.19, ”Disgree”: 0.20-0.39, ”Neutral”: 0.40-0.59, ”Agree”: 0.60-0.79, ”Strongly agree”: 0.80-1). The results were then analysed and were compared to the numbers produced by the algorithm.

The pairs of the individual learning outcomes were evaluated by the group of 38 Russian students who had previously studied the corresponding modules. In four cases out of five, the results produced by the algorithm fell within the intervals with the maximum number of replies. In the last case, the answers were almost equally distributed between the ”Disagree” and ”Agree” answers, while the algorithm returned the value in the ”Neutral” interval.

The first group of modules was compared by two groups of experts. The first consisted of academic staff, PhD students and IT business representatives, while the second included undergraduate students. With the help of the Mann-Whitney U test, we proved that the results produced by the different groups could be amalgamated. The second group of modules was evaluated by the first group of experts only. The results showed that the algorithm for the comparison of the keywords tended to produce more positive results (”Agree”) than negative (”Disagree”) ones, coinciding with the experts' answers in three cases out of five. At the same time, the algorithm for the comparison of the learning outcomes followed the respondents' opinion in four experiments out of five, thus reflecting appropriate agreement/disagreement tendencies. The overall estimation was correct in four cases out of five. On balance, the proposed approach and the designated algorithm can be used to compare educational courses and modules and will produce reasonable results.

Chapter 7

Conclusions and Future Work

In order to conclude this research work, we need to summarise what has been done and to assess the results achieved. Possible directions for the further development of this work will also be mentioned.

7.1 Research Summary

The research consisted of the following main stages.

Firstly, the current situation of the research topic in both of the fields under investigation was carefully analysed. The findings provided the groundwork on which to base the comparison of the modules and courses on the keywords and learning outcomes. Ontologies were considered to be an applicable apparatus for the data representation. The ”Keywords” and ”Learning Outcomes” sections of the module templates from the De Montfort and Bauman Moscow State Technical Universities were studied in detail. As a result, the ontology of a course, module, keywords and learning outcomes was created. The set of SWRL-rules was produced to simplify the ontology population. The ontology was implemented manually via the application Protégé 4.0.2.

In order to partially automate the population of the ontologies, we decided to design an application that would annotate the texts with tags, coinciding with the names of the ontology entities. To achieve this goal, the typical structures of single keywords and learning outcomes were examined. As a result, the grammars of the

keywords and learning outcomes were created. The annotators of the keywords and the learning outcomes were implemented using the General Architecture for Text Engineering 6.1.

We decided to create a novel similarity measure, which would allow us to distinguish the comparison of the learning outcomes from the comparison of any other sentences. The aim was to introduce a semantic measure that would characterise the similarities in the educational domain. The investigation of educational theory was suggested by B. Bloom's taxonomy of the educational aims revised by D. Krathwohl. Based on this, we designed a similarity measure for the action verbs and called it CAVe (**c**ognitive space's semantic similarity between the **a**ction **ve**rbs).

The ontology alignment algorithm was based on the findings of the previous work packages. It utilises the ontology of a course and a module, including the sub-ontologies of a learning outcome and data domain, via the similarity measure CAVe. In addition to CAVe, we also used the Levenshtein edit distance, Soundex, Wu and Palmer and the DISCO2 similarity measures. The CAVe similarity measure and the ontology alignment algorithm were implemented in Java.

In order to evaluate the designed system, we used the module templates from the De Montfort and Bauman Moscow State Technical Universities. The Russian documents were manually translated into English.

The chosen module templates were annotated using the GATE applications. The results were transmitted to the ontologies. The quality of annotating was evaluated against manually created gold standards with the help of the AnnotationDiff tool provided by the GATE 6.1. High values of precision, recall and f-measure proved the admirable quality of the annotating.

In order to evaluate the alignment algorithm, we created two packages of short module templates. The first contained the modules in the field of data modelling and databases. The second consisted of the modules concerned with object-oriented programming. The short module templates, which contained only the sections "Keywords" and "Learning outcomes", were designed. In each of the groups, one master module was chosen, against which all the others were compared. The alignment algorithm covered the pairs of the modules' ontologies.

The questionnaire, which contained the short module templates, was prepared. The respondents were asked to use the Likert scale to determine if they "Strongly

disagreed", "Disagreed", felt "Neutral", "Agreed" or "Strongly agreed" that each pair of the modules was similar regarding the keywords and the learning outcomes, as well as being similar overall.

Two groups of experts were chosen. The first consisted of academic staff and PhD students from Europe, as well as people working in the IT business in Russia. The second group consisted of undergraduate Russian students who had just completed modules on data models and databases. Their answers to the questionnaire were received and were placed in an .xls file.

The results of the algorithm were compared to those produced by human judgement, using the statistical approach. We used descriptive statistics (mode, median) and inferential statistics (the Mann-Whitney U test). The evaluation showed that the algorithm coped well with the pairwise and group comparison of the learning outcomes. At the same time, when the respondents disagreed regarding the similarity of the keywords, it returned values that were too high. The overall estimate of the similarity of the modules provided by the algorithm coincided with the opinions of the human respondents.

7.2 Revisiting the Research Questions and Contributions

The following main research question was formulated in Section 1.3.

How to automate the comparison of higher education courses and modules, belonging to cognitive learning domain, using a document and ontology-based approach?

The main research contribution of the research answered this question. This main contribution is the novel methodology for the automated comparison of academic courses and modules, belonging to cognitive learning domain, based on ontology alignment. The methodology and its implementation are presented in detail in Chapters 4, 5.

The other research questions and corresponding contributions are discussed below.

1. Which information regarding educational courses and modules should be used for comparison and how should it be stored in an ontology?

In order to answer this question, we analysed the recommendations produced by the Bologna Workgroup, the universities' programme specifications and the module templates (see Chapter 2 for details).

The programme specifications contain the title, the educational level, information regarding the university/ faculty/ department, competences and the list of the course's modules. The module templates contain the following key information about the educational modules: the titles, the number of credits, the keywords and the learning outcomes. Special significance is attached to the learning outcomes, which are "statements of what a learner is expected to know, understand and/or be able to demonstrate after completion of a process of learning" [41]. In this way, they summarise how teaching, learning and assessment strategies lead to the achievement of the educational goal. The keywords are important, because they provide an indication of the main data domain concepts touched upon and learnt during the educational process.

The ability to represent the semantics formally and to make inferences was the key requirement for the data model. For this reason, the programme specifications and the module templates were represented as ontologies.

We designed a reusable ontology for a course and for a module, including the data domain and the learning outcome sub-ontologies. In brief, the course and module section stores general information concerning them. A course is addressed as a set of modules; thus, course-learning outcomes are inherited from the modules' LOs using SWRL-rules. The sub-ontology of a learning outcome addresses the main clause of a LO as a tuple $< ActionVerbConstruction, DataDomainObjectConstruction >$, which reflects its real life structure in the natural language. Here, the Action Verb Construction defines the operation, which a student should be able to perform via the Data Domain Object Construction. The data domain sub-ontology stores the keywords of the topics taught in the modules. All of the ontology's parts are populated from the programme specifications and the module templates, and are connected to each other through particular relations.

This ontology is the first minor contribution of this work.

2. How to automate the population of the ontology with the data from the documents?

Ontology population from the documents may be automated if the texts are annotated with tags that coincide with the corresponding ontology entities.

In order to populate the ontologies, we created two applications in GATE. These implemented the grammars of the keywords and the learning outcomes. Both grammars were designed and implemented as part of this research work. The applications received the short module templates, containing the keywords and the learning outcomes, as inputs. They annotated the texts with the tags, which later allowed us to place the marked-up pieces into the ontology classes according to their instances.

In order to evaluate the quality of the annotating, we utilised the GATE tool, AnnotationDiff. This allows for the calculation of precision, recall and F-measure by comparing the annotations produced by the application against the gold standard. In our case, the gold standard annotations were created manually. The high values of precision, recall and F-measure showed that the annotators performed well in annotating both the ”Keywords” and the ”Learning Outcomes” sections. Therefore, we can recommend using them in the future without the need for amendments.

The formal grammars and the annotators of the keywords and of the learning outcomes constitute the second minor contribution of this research.

3. What is the alignment algorithm for the ontologies of educational courses and modules? Which similarity measures should it utilise?

The educational courses and the modules were compared by aligning the ontologies of their keywords and their learning outcomes. The algorithm is presented in detail in Chapter 4. It utilises combinations of different similarity measures, including the Levenshtein edit distance, Soundex, Jaro-Winkler, Wu and Palmer and the DISCO2.

A special semantic measure, which allows for the comparison of the learning outcomes as applied to educational domain, was designed. The measure, CAVe, was designed and implemented for the comparison of the action verbs. It is based on the distance between the verbs in B. Bloom’s revised taxonomy of educational aims.

The alignment algorithm was implemented in Java. The details are presented in Chapter 5.

We could not find any existing automated systems for the comparison of the educational modules, their keywords and learning outcomes. Thus, we compared the algorithm's results to those generated by human judgement. The details are presented in Chapter 6. In order to evaluate the results returned by the algorithm, we used it to compare a number of the module templates. Afterwards, a number of experts were asked to evaluate the similarity between these modules based on their keywords and learning outcomes. The results were compared using the statistical approach.

The algorithm for the comparison of the learning outcomes performed well in both cases in which it was tested, namely for the pairs of single learning outcomes and for all the learning outcomes of the modules from the corresponding sections. The algorithm for the comparison of the sets of keywords performed well in the cases when a high similarity value was to be achieved, but returned similarity values that were too large when it was possible to reject the similarity between the modules' keywords. The overall performance of the algorithm was good.

The similarity measure, CAVe, and the ontology alignment algorithm are the third and fourth minor contributions of this research, respectively.

7.3 The Scope of Applicability and Limitations

As mentioned in Section 1.2, the designed methodology is aimed at the comparison of educational courses and modules belonging to the cognitive learning domain only, and is not applicable for courses and modules in the affective or in the psychomotor domain.

Our approach does not consider cross-cultural aspects that could possibly be related to the comparison of educational courses and modules. We claim that our approach is applicable for the universities sharing the definitions of the programme specifications, module templates and learning outcomes presented in Chapter 2.

The current implementation of the methodology was created to compare the module templates and programme specifications written in the English language only. Nevertheless, the entire methodology can be transferred to modules and courses in other languages, provided that the ideas of keywords and learning outcomes

are preserved. To do this, one should create formal grammars of the learning outcomes and keywords in the desired language to produce corresponding annotators. The similarity CAVe measure can be computed for two verbs in any language, as long as the ontology of B. Bloom's revised taxonomy is translated into the said language.

7.4 Revisiting the Research Methodology

The research was carried out in accordance with the constructive research methodology, as was indicated in Section 1.4. This approach guided us through the problem statement, the literature review and the design, implementation and evaluation phases of the research work. As a result, we answered the research questions and achieved one major and four minor research contributions. The stages of the constructive research methodology were addressed in the following way.

Firstly, we chose a practical, relevant topic that had research potential. The automation of the comparison of educational courses and modules is an appropriate topic. The results of the comparison can be used for such tasks as the creation of joint educational programmes. Through this application, they may ultimately be useful for the achievement of the main goals of the Bologna process, which include developing the collaboration among universities and encouraging students' mobility. This topic provided us with vast research opportunities in the fields of education and computer science. We studied ways in which educational courses and modules are currently compared, as well as the information used in this process. We then researched computer science methods in order to solve the problem of process automation. This literature review was the second stage of the research methodology.

The following conclusions were reached, based on the literature review produced as a result of the document-based research. The educational modules should be compared based on their keywords and learning outcomes, as these are their key parameters. An educational course may be treated as a set of its modules. An educational course, module, their keywords and learning outcomes can be presented in the form of an ontology.

In the third stage of the methodology, we developed a methodology for the comparison of educational courses and modules based on ontology alignment.

The modelling research method helped us to identify the main concepts and relationships between them. The set-theoretic model was converted to the ontology for the formal representation of an educational course, its modules, keywords and learning outcomes. Formal grammars were used to model the keywords and learning outcomes. They helped to create the annotators for the partially automatic population of the ontology.

Similarity measures theory was used to create the similarity measure CAVe for the action verbs of the learning outcomes, based on B. Bloom's revised taxonomy of educational objectives. The ontology alignment algorithm for the educational modules and courses was based on the theoretical ontology alignment process, as presented in Chapter 2.

The fourth stage of the research methodology is presented in Chapters 5 and 6.

We implemented the similarity measure, CAVe, and the ontology alignment algorithm using the object-oriented design method in Java. This gave us the opportunity to utilise already existing APIs to process the ontologies and similarity measures.

The evaluation of the quality of the annotating and of the alignment algorithm was performed with the help of qualitative and quantitative methods.

The quality of the annotations was evaluated with the help of standard methodology. The annotations provided by the recognisers were compared to the manually created gold standard. Afterwards, the precision, recall and f-measure were computed. This evaluation methodology was beneficial for the following reasons. Firstly, this is a typical method used to evaluate the annotators; thus, other researchers can easily interpret the evaluation results. Secondly, this is a quantitative approach, which excludes the researcher's bias. Thirdly, the results of this evaluation can be easily reproduced.

In order to evaluate the performance of an algorithm, we chose several modules from the De Montfort and Bauman Moscow State Technical Universities. Five case studies, containing pairs of short module templates, were created. We asked the experts to fill in a questionnaire by answering if they "Strongly disagreed",

"Disagreed", stayed "Neutral", "Agreed" or "Strongly agreed" that the two modules were similar in terms of to their keywords and learning outcomes, as well as being similar overall. A five-point Likert scale was used to collect the replies. The intervals of values returned by the alignment algorithm were assigned to each of the possible human answers. Afterwards, we compared the algorithm's performance to the experts' opinions using statistical methods. This evaluation methodology was chosen due to the nature of the research. We aimed to create a methodology that would return the same results for the comparison of the courses and modules as those given by a human expert. The results of the evaluation led us to conclude that our algorithm reflected the tendencies of the human answers in terms of agreeing or disagreeing on the similarity of the modules. Moreover, we discovered that the algorithm returned the values in the "Neutral" interval, when almost equal numbers of the experts' votes were divided between the agreement and disagreement categories of answers.

The fifth and sixth stages of the methodology are addressed as follows.

The theoretical connections of the research are based on the literature review provided in Chapter 2. Their analysis helped us to position our interdisciplinary research within the fields of education and computer science. The research contributions are discussed in Chapters 1 and 7. The scope of applicability is discussed in Section 7.3 of the current chapter.

7.5 Future Work

Future work on this project includes the following directions.

Firstly, we plan to improve current realisation of the methodology. We would like to make the ontology population fully automatic. In order to carry out the experiments, we populated the ontologies manually, based on the annotations of keywords and learning outcomes produced by the corresponding GATE applications. In the future, we plan to create a standalone application, which will input the ontology entities into the ontologies automatically, based on the annotations. The aggregation strategy for the "Keywords" section also needs some refinement. We need to improve the aggregation strategy for the modules' keywords.

In the future, we would also like to investigate the following research problems.

We wish to improve the comparison of courses containing many modules. Currently, the alignment algorithm performs a pairwise comparison of all the keywords and learning outcomes of the educational modules and courses. This approach can be time consuming when applied to large educational programmes. We plan to improve this by creating an approach that clusters the keywords and learning outcomes. In this case, we would be able to choose one keyword or learning outcome from each cluster and use only those for comparison. This would allow us to reduce the space and time needed for the computation of the similarity.

The current algorithm is aimed at the computation of the similarity of the intersection of the courses' and modules' keywords and learning outcomes. We would also like to explore the oriented similarity, which is based on a subsumption relationship. This approach would be useful for cases in which it is necessary to understand if one module or course would be subsumed by another.

Last, but not the least, we would like to test the designed methodology using disciplines and courses from various fields of studies in the cognitive learning domain.

7.6 Concluding Remarks

The results of the research conducted in this work are timely for such tasks as prior learning and degree recognition, the introduction of joint educational programmes and quality assurance in higher education. They can be used to facilitate the creation of the European Higher Education Area as defined in the Bologna Declaration. It is obvious that the algorithm cannot fully substitute for highly qualified experts; however, it can be used to reject dissimilar modules and to outline the templates that are extremely likely to be similar, or which require deeper analysis.

Bibliography

[1] Oxford dictionaries. Accessed on 12 August 2013. [Online]. Available: http://www.oxforddictionaries.com/

[2] *Joint Declaration on Harmonisation of the Architecture of the European Higher Education System*, Std., 1998.

[3] *Joint Declaration of the European Ministers of Education*, Std., 1999.

[4] (2007, May) The Soundex Indexing System. Accessed on 12 August 2013. [Online]. Available: http://www.archives.gov/research/census/soundex.html

[5] (2011) National Occupational Standards. Accessed on 15 March 2013. [Online]. Available: http://www.ukstandards.co.uk/about-nos/Pages/About-NOS.aspx

[6] (2011) Revision of the International Standard Classification of Education (ISCED). Accessed on 16 May 2013. [Online]. Available: http://www.uis.unesco.org/Education/Documents/isced-2011-en.pdf

[7] *Altova SemanticWorks 2012 User and Reference Manual*, 2012.

[8] D. Allemang and J. Hendler, *Semantic Web for the Working Ontologist Modeling in RDF, RDFS and OWL*. Morgan Kaufmann Publishers, 2008.

[9] S. Amrouch and S. Mostefai, "Syntactico-semantic algorithm for automatic ontology merging," in *Proceeding of the International Conference on Information Technology and e-Services*, March 2012, pp. 1–5.

[10] L. Anderson, D. Krathwohl, and B. Bloom, *A Taxonomy for Learning, Teaching, and Assessing: a Revision of Bloom's Taxonomy of Educational Objectives*. Longman, 2001.

[11] J. Antony and F. Thottupuram, "Semantic web based adaptive e-learning triggered through short message services," in *Proceedings of the 7th International Conference on Computer Science and Education*, July 2012, pp. 1860–1863.

[12] J. C. Arparez, O. Corcho, M. Fernandez-Lopez, and A. Gomez-Perez, "WebODE: a scalable workbench for ontological engineering," in *Proceedings of the First International Conference on Knowledge Capture (K-CAP)*, 2001, pp. 21–23.

[13] A. Arwan, "Ontology and semantic matching for diabetic food recommendations," in *Proceedings of the International Conference on Information Technology and Electrical Engineering*, October 2013, pp. 170–175.

[14] F. Baader, D. McGuinness, D. NardiI, and P. Schneider, *The Description Logic Handbook: Theory, implementation, and applications*. Cambridge University Press, 2003.

[15] F. Baader, I. Horrocks, and U. Sattler, "Description Logics as Ontology Languages for the Semantic Web," in *Festschrift in honor of Jorg Siekmann, Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2003, pp. 228–248.

[16] Bashmakov A. I. and Bashmakov I.A., *Intelligent Information Technologies*. BMSTU Publishing House, 2005.

[17] *Sample Main Educational Programme for the Profile 230110 Computer Science and Engineering (qualification Bachelor of Science)*, Bauman Moscow State Technical University Std., 2009.

[18] D. Bertram, "Likert Scales," Tech. Rep.

[19] E. Blanchard, M. Harzallah, H. Briand, and P. Kuntz, "A Typology of Ontology-based Semantic Measures," in *Proceedings of the Enterprise Modelling and Ontologies for Interoperability*, vol. 160, June 2005, pp. 13–14.

[20] B. Bloom, *Taxonomy of Education Objectives*. Longmans Green and Company, 1956.

[21] *Using Learning Outcomes*, Bologna Working Group Std., 2004.

[22] *A Framework for Qualifications of the European Higher Education Area*, Bologna Working Group on Qualifications Frameworks Std., 2005.

[23] K. Bontcheva, V. Tablan, D. Maynard, and H. Cunningham, "Evolving GATE to Meet New Challenges in Language Engineering," *Natural Language Engineering*, vol. 10, no. 3/4, pp. 349–373, 2004.

[24] A. Bryman, *Social Research Methods*. Oxford University Press, Incorporated, 2004.

[25] D. Calvanese, J. Carroll, G. D. Giacomo, and J. Hendler. (2012, December) Owl 2 web ontology language profiles (second edition). Accessed on 15 August 2013. [Online]. Available: http://www.w3.org/TR/owl2-profiles/

[26] A. Cartelli, "Semantics, Ontologies and Information Systems in Education: Concerns and Proposals," *Issues in Informing Science and Information Technology*, vol. 3, pp. 113–125, 2006.

[27] P. Chahal, M. Singh, and S. Kumar, "Ranking of web documents using semantic similarity," in *Proceedings of the International Conference on Information Systems and Computer Networks*, 2013, pp. 145–150.

[28] M. Cheatham. (2011) MapSSS Results for OAEI 2011. Accessed on 15 August 2013. [Online]. Available: http://ceur-ws.org/Vol-814/oaei11_paper11.pdf

[29] N. Choi, I.-Y. Song, and H. Han, "Survey on Ontology Mapping," *SIGMOD Record*, vol. 35, no. 3, pp. 34–41, 2006.

[30] O. Corcho. (2002) OntoWeb: Ontology-based Information Exchange for Knowledge Management and Electronic Commerce. Accessed on 12 July 2013. [Online]. Available: http://www.sti-innsbruck.at/sites/default/files/D11.1_0.pdf

[31] V. Cross and A. Chennai-Thiagarajan, "Measuring information content for an ontological concept," in *Annual Meeting of the North American Fuzzy Information Processing Society*, 2012, pp. 1–6.

[32] V. V. Cross, P. Silwal, and X. Chen, "Semantic similarity measures in ontology alignment," in *IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS)*, June 2013, pp. 442–447.

[33] I. Cruz, C. Stroe, F. Caimi, A. Fabiani, and C. Pesquita. (2011) Using AgreementMaker to Align Ontologies for OAEI 2011. Accessed on 12 July 2013. [Online]. Available: http://ceur-ws.org/Vol-814/oaei11_paper1.pdf

[34] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan, "GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications," in *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, 2002.

[35] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, N. Aswani, I. Roberts, G. Gorrell, A. Funk, A. Roberts, D. Damljanovic, T. Heitz, M. A. Greenwood, H. Saggion, J. Petrak, Y. Li, and W. Peters, *Text Processing with GATE (Version 6)*, 2011.

[36] ——, *Developing Language Processing Components with GATE version 7 (a User Guide)*, The University of Sheffield, Department of Computer Science, Regent Court 211 Portobello Sheffield S1 4DP United Kingdom, 2012.

[37] J. C. de Winter and D. Dodou, "Five-Point Likert Items: t test versus Mann-Whitney-Wilcoxon," *Practical Assessment, Research & Evaluation*, no. 11, October 2012.

[38] G. Dodig-Crnkovic, "Constructivist Research and Info-Computational Knowledge Generation," in *Model-Based Reasoning In Science And Technology - Abduction, Logic, and Computational Discovery (Studies in Computational Intelligence)* . Springer Verlag , 2010, pp. 359–380.

[39] M. Ehrig, *Ontology Alignment: Bridging the Semantic Gap*. Springer, 2007.

[40] H. Elasri and A. Sekkaki, "Background ontology used in ontologies alignment to support integration process of business components," *Applied Mathematical Sciences*, vol. 7, pp. 969–981, 2013.

[41] *ECTS Users' Guide*, European Commission Std., 2009.

[42] J. Euzenat and P. Shvaiko, *Ontology Matching*. Heidelberg (DE): Springer-Verlag, 2007.

[43] B. Glimm, M. Horridge, B. Parsia, and P. F. Patel-Schneider, "A Syntax for Rules in OWL 2," in *OWL: Experiences and Directions (OWLED2009)*, R. Hoekstra and P. F. Patel-Schneider, Eds., 2009.

[44] T. Gruber, "Principles for the Design of Ontologies Used for Knowledge Sharing," *International Journal of Human and Computer Studies*, vol. 5/6, no. 43, pp. 907–928, 1993.

[45] N. Guarino, "OntoSeek: Content-Based Access to the Web," *IEEE Intelligent Systems*, pp. 70–80, 1999.

[46] J. Hebeler, M. Fisher, R. Blace, and A. Perez-Lopez, *Semantic Web Programming*, M. Dean and M. Smith, Eds. Wiley Publishing, Inc., 2009.

[47] S. R. Heiyanthuduwage, R. Schwitter, and M. A. Orgun, "An adaptive learning system using plug and play ontologies," in *Proceedings of IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, 2013, pp. 429–434.

[48] P. Hitzler, M. Krotzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph. (2009, October) OWL 2 Web Ontology Language Primer. Accessed on 15 August 2013. [Online]. Available: http://www.w3.org/TR/2009/REC-owl2-primer-20091027/#OWL_2_DL_and_OWL_2_Full

[49] M. Hoffmann, "Using Blooms Taxonomy of Learning to Make Engineering Courses Comparable," in *EAEEIE Annual Conference*, 2008.

[50] M. Horridge, S. Jupp, G. Moulton, A. Rector, R. Stevens, and C. Wroe, "A Practical Guide To Building OWL Ontologies Using Protege 4 and CO-ODE Tools," The University Of Manchester, Tech. Rep., 2007.

[51] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. (2004, May) SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Accessed on 12 June 2013. [Online]. Available: http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/

[52] I. Horrocks, "DAML+OIL: a Description Logic for the Semantic Web," *Bull. of the IEEE Computer Society Technical Committee on Data Engineering*, vol. 25, pp. 4–9, 2002.

[53] ——, "Description logics in ontology applications," in *Proceedings of the 14th international conference on Automated Reasoning with Analytic Tableaux and Related Methods*, ser. TABLEAUX'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 2–13. [Online]. Available: http://dx.doi.org/10.1007/11554554_2

[54] I. Horrocks, O. Kutz, and U. Sattler, "The even more irresistible SROIQ," in *International Conference on the Principles of Knowledge Representation and Reasoning*. AAAI Press, 2006, pp. 57–67.

[55] I. Horrocks, B. Parsia, and U. Sattler. (2009, October) OWL 2 Web Ontology Language Direct Semantics. Accessed on 16 August 2013. [Online]. Available: http://www.w3.org/TR/2009/REC-owl2-direct-semantics-20091027/

[56] I. Horrocks, P. F. Patel-Schneider, D. L. McGuinness, and C. A. Welty, "OWL: a Description Logic Based Ontology Language for the Semantic Web," in *The Description Logic Handbook: Theory, Implementation, and Applications (2nd Edition)*. Cambridge University Press, 2007.

[57] L. Huang, "Review of ontology mapping," in *The Proceedings of the 2nd International Conference on Consumer Electronics, Communications and Networks*, 2012, pp. 537–540.

[58] J. Huber, T. Sztyler, J. Noessner, and C. Meilicke, "CODI: Combinatorial Optimization for Data Integration Results for OAEI 2011," in *Ontology Matching OM-2011*, 2011.

[59] S. Hussain, J. D. Roo, and M.-C. Jaulent, "Proof-based ontology matching: Finding semantic similarities between ancestor graph structures," in *Proceedings of IEEE Sixth International Conference on Semantic Computing*, 2012, pp. 202–209.

[60] S. Jamieson, "Likert Scales:How to (Ab)Use Them," *Medical Education*, vol. 38, pp. 1217–1218, 2004.

[61] *Shared Dublin descriptors for Short Cycle, First Cycle, Second Cycle and Third Cycle Awards*, Joint Quality Initiative Std., 2004.

[62] I. L. Kaftannikov, "The Perspectives to Use Web-ontologies in Educational Process," *Educational Technology & Society*, vol. 6(3), p. 1340138, 2003.

[63] Y. Kalfoglou and M. Schorlemmer, "Ontology mapping: the state of the art," *The Knowledge Engineering Review*, vol. 18(1), pp. 1–31, 2003.

[64] D. Kanellopoulos, "Ontology-based Learning Applications: a Development Methodology," in *Proceedings of the 24th IASTED International Multi-Conference*, 2006.

[65] B. Karaoglan and T. Kisla, "Course prescription using ontologies," in *Proceedings of the 24th EAEEIE Annual Conference*, May 2013, pp. 184–186.

[66] M. Keane. (2009, April) Guide to writing module learning outcomes at dcu. Accessed on 3 August 2013. [Online]. Available: http://www.dcu.ie/afi/docs/FINAL_GUIDE_LOs-1%20May%2019th.pdf

[67] D. Kennedy, A. Hylan, and N. Ryan, *Writing and Using Learning Outcomes: a Practical Guide*, University College Cork, 2007.

[68] J. Kim, P. Shin, and H. Chung, "TV Program Search and Recommendation Based on TV and Viewer Ontologies," in *Proccedings of the International Conference on Information Science and Applications*, June 2013.

[69] S. Knittl and D. Schmitz, "An ontology-based approach for semantic interoperability in interorganizational it service management." in *IM*, F. D. Turck, Y. Diao, C. S. Hong, D. Medhi, and R. Sadre, Eds. IEEE, 2013, pp. 1218–1224.

[70] P. Kolb, "DISCO: A Multilingual Database of Distributionally Similar Words," in *KONVENS 2008 - Erganzungsband: Textressourcen und lexikalisches Wissen*, A. S. A. Storrer, A. Geyken and K.-M. Wurzner, Eds., 2008, pp. 37–44.

[71] ——, "Experiments on the Difference between Semantic Similarity and Relatedness," in *Proceedings of the 17th Nordic Conference of Computational Linguistics NODALIDA 2009*, K. Jokinen and E. Bick, Eds. Northern European Association for Language Technology, 2009, pp. 81–88.

[72] D. R. Krathwohl, "A Revision to Bloom's Taxonomy: Overview," *Theory into Practice*, vol. 41, no. 4, 2002.

[73] M. Krotzsch, F. Simancik, and I. Horrocks, "A Description Logic Primer," *CoRR*, vol. abs/1201.4089, 2012.

[74] O. Lassila and R. Swick, *Resource Description Framework (RDF) Model and Syntax Specification*, W3C, 1999.

[75] M. Lemnaru, M. Dobrin, M. Florea, and R. Potolea, "Designing a travel recommendation system using case-based reasoning and domain ontology," in *Proceedings of the IEEE International Conference on Intelligent Computer Communication and Processing*, September 2012, pp. 23–30.

[76] D. Lin, "An Information-theoretic Definition of Similarity," in *Proceedings of the 15th International Conference on Machine Learning*, 1998, pp. 296–304.

[77] R. Lowry. (2013) Concepts & Applications of Inferential Statistics. Accessed on 12 May 2013. [Online]. Available: http://vassarstats.net/textbook/index.html

[78] Lukka, K. and Reponen, T., "The Key Issues of Applying the Constructive Approach to Field Research," in *Management Expertise for the New Millenium. In Commemoration of the 50th Anniversary of the Turku School of Economics and Business Administration.* , T. Reponen, Ed. Publications of the Turku School of Economics and Business Administration, 2000, p. 113.

[79] T. G. M. Ushold, "Ontologies: Principles, Methods and Applications," *Knowledge Engineering Review*, vol. 11, no. 2, 1996.

[80] S. J. Mann. (2004) Guidelines for writing aims and intended learning outcomes at the programme and course level. Accessed on 5 August 2013. [Online]. Available: http://www.gla.ac.uk/media/media_105307_en.pdf

[81] J. Mesaric and B. Dukie, "An Approach to Creating Domain Ontologies for Higher Education in Economics," in *Proceedings of the ITI 29th Interntational Conference on Information Technology Interfaces*, 2007, pp. 75–80.

[82] M. Milicka and R. Burget, "Web document description based on ontologies," in *Proceedings of the Second International Conference on Informatics and Applications*, 2013, pp. 288–293.

[83] F. Miller and J. M. A. Vandome, *Likert Scale*. VDM Publishing, 2010.

[84] G. Miller, "Wordnet an On-line Lexical Database," *International Journal of Lexicography*, vol. 3(4), pp. 235–312, 1990.

[85] ——, "WordNet: A Lexical Database for EnglishWordNet: A Lexical Database for English," in *Proceedings of Commun. ACM*, 1995, pp. 39–41.

[86] G. Miller and F. Hristea, "WordNet Nouns: Classes and Instances," in *Proceedings of Computational Linguistics*, 2006, pp. 1–3.

[87] *Federal State Educational Standard of Higher Professional Education for the profile 230100 Computer Science and Engineering (qualification Bachelor of Science)*, Ministry of Education and Science of the Russian Federation Std., 2009.

[88] *Federal State Educational Standard of Higher Professional Education for the profile 230100 Computer Science and Engineering (qualification Master of Science)*, Ministry of Education and Science of the Russian Federation Std., 2009.

[89] *Sample Main Educational Programme for the Profile 010040 Applied Mathematics and Computer Science (qualification Bachelor of Science)*, Moscow State University after M.V. Lomonosov Std., 2010.

[90] D. Ngo, Z. Bellasene, and R. Coletta. (2011) YAM++ Results for OAEI 2011. Accessed on 12 May 2013. [Online]. Available: http://ceur-ws.org/Vol-1111/oaei13_paper16.pdf

[91] J. Niekerk and R. Solms, "Using Bloom's Taxonomy for Information Security Education," in *Information Assurance and Security Education and Training*, ser. IFIP Advances in Information and Communication Technology, J. Dodge, RonaldC. and L. Futcher, Eds. Springer Berlin Heidelberg, 2013, vol. 406, pp. 280–287.

[92] N. F. Noy and D. L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology," Stanford University, Tech. Rep., 2001.

[93] B. J. Oates, *Researching Information Systems and Computing*. SAGE, 2006.

[94] O.Luimnigh. (2007) WritingLearningOutcomes. Accessed on 1 August 2013. [Online]. Available: http://www.fibaa.org/uploads/media/Writing_Learning_Outcomes_at_Programme_and_Module_Levels.pdf

[95] G. Pirró, "A Semantic Similarity Metric Combining Features and Intrinsic Information Content," *Data Knowl. Eng.*, vol. 68, no. 11, pp. 1289–1308, 2009.

[96] N. Prat, J. Akoka, and I. Comyn-Wattiau, "Transforming multidimensional models into OWL-DL ontologies," in *The Proceeedings of the Sixth International Conference on Research Challenges in Information Science*, May 2012, pp. 1–12.

[97] *Guidelines for Preparing Programme Specifications*, The Quality Assurance Agency for Higher Education Std., 2006.

[98] *Handbook for Institutional audit: England and Northern Ireland*, The Quality Assurance Agency for Higher Education Std., 2009.

[99] M. Ramprasath and S. Hariharan, "Using ontology for measuring semantic similarity for question answering system," in *Proceedings of the IEEE International Conference on Advanced Communication Control and Computing Technologies*, 2012, pp. 218–223.

[100] P. Resnik, "Using Information Content to Evaluate Semantic Similarity in a Taxonomy," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, vol. 1, 1995, pp. 448–453.

[101] J. Roy and A. Guyard, "Supporting threat analysis through description logic reasoning," in *Proceedings of the IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support*, 2012, pp. 308–315.

[102] A.-B. Sarma Cakula and M. Salem, "E-learning developing using ontological engineering," *WSEAS Transactions on Information Science and Applications*, vol. 10, pp. 14–25, 2013.

[103] P. Shvaiko and J. Euzenat, "Ontology matching: State of the art and future challenges," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, pp. 158–176, 2013.

[104] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: a Practical OWL-DL Reasoner," *Web Semant.*, vol. 5, no. 2, pp. 51–53, Jun. 2007.

[105] M. Smith, C. Welty, and D. McGuinness. (2004, February) OWL Web Ontology LanguageGuide. Accessed on 12 May 2013. [Online]. Available: http://www.w3.org/TR/2004/REC-owl-guide-20040210/#OwlVarieties

[106] S. Sosnovsky, "Development of Educational Ontology for C-programming," *Information Theories and Applications*, vol. 13, pp. 303–308, 2005.

[107] A. Souag, C. Salinesi, and I. Wattiau, "Using security and domain ontologies for security requirements analysis," in *Proceedings of IEEE 37th Annual Computer Software and Applications Conference Workshops*, 2013, pp. 101–107.

[108] N. T. Takeda H., Takaai M., "Collaborative Development and Use of Ontologies for Design," in *Proceedings of the Tenth International IFIP WG 5.2/5.3 Conference PROLAMAT 98*, 1998.

[109] J. Tauberer. Quick Intro to RDF. Accessed on 17 April 2010. [Online]. Available: http://www.rdfabout.com/quickintro.xpd

[110] Q.-V. Tran, R. Ichise, and B-Q.Ho. (2011) Cluster-based Similarity Aggreagtion for Ontology Matching. Accessed on 15 June 2013. [Online]. Available: http://ceur-ws.org/Vol-814/oaei11_paper5.pdf

[111] D. Tsarkov and I. Horrocks, "DL Reasoner vs. First-Order Prover," in *Proc. of the 2003 Description Logic Workshop*, 2003, pp. 152–159.

[112] (2011) Code of Practice. UK NARIC. Accessed on 12 July 2013. [Online]. Available: http://www.naric.org.uk/index.asp?page=9

[113] *Convention on the Recognition of Qualifications concerning Higher Education in the European Region*, UNESCO Std., 1997.

[114] (2013, August) Revision of the International Standard Classification of Education: Fileds of Education and Training (ISCED-F). UNESCO. Accessed on 15 February 2014. [Online]. Available: http://www.uis.unesco.org/Education/Documents/isced-37c-fos-review-222729e.pdf

[115] P. Viana and J. Becerra, "Applying ontological similarity to automatic service composition to generate alternative business processes," in *Proceedings of the Sixth Brazilian Symposium on Software Components Architectures and Reuse*, September 2012, pp. 111–119.

[116] A. J. Wiebe and C. W. Chan, "Ontology driven software engineering electrical and computer engineering," in *Proceedings of the 25th IEEE Canadian Conference on*, 2012, pp. 1–4.

[117] Z. Wu and M. Palmer, "Verb Semantics and Lexical Selection," in *Proceedings of the FQAS ACL-94*, 1994, pp. 133–138.

[118] H. Xiao, T. Qiu, and P. Zhou, "Integration of heterogeneous agriculture information system based on interoperation of domain ontology," in *Proceedings of the Second International Conference on Agro-Geoinformatics*, August 2013, pp. 476–480.

[119] J.-M. Yves, E. Shironoshita, and M. Kabuka. (2009) ASMOV: Results for OAEI 2009. Accessed on 12 May 2013. [Online]. Available: http://ceur-ws.org/Vol-551/oaei09_paper4.pdf

[120] L. Zemmouchi-Ghomari and A. R. Ghomari, "Process of building reference ontology for higher education," in *Proceedings of the World Congress on Engineering*, vol. 3, 2013, pp. 1595–1600.

Appendix A

The Taxonomies of Educational Objectives

The original taxonomy by B. Bloom has the following structure

1. Knowledge
   (a) Knowledge of specifics;
   (b) Knowledge of terminology;
   (c) Knowledge of specific facts;
   (d) Knowledge of ways and means of dealing with specifics;
   (e) Knowledge of conventions;
   (f) Knowledge of trends and sequences;
   (g) Knowledge of classifications and categories;
   (h) Knowledge of criteria;
   (i) Knowledge of methodology;
   (j) Knowledge of universals and abstractions in a field;
   (k) Knowledge of principles and generalizations;
   (l) Knowledge of theories and structures.
2. Comprehension

   (a) Translation;
   (b) Interpretation;
   (c) Extrapolation.

3. Application
4. Analysis
   (a) Analysis of elements;
   (b) Analysis of relationships;
   (c) Analysis of organizational principles.
5. Synthesis
   (a) Production of a unique communication;
   (b) Production of a plan, or proposed set of operations;
   (c) Derivation of a set of abstract relations.
6. Evaluation
   (a) Evaluation in terms of internal evidence;
   (b) Judgments in terms of external criteria.

Structure of the Knowledge Dimension of the Revised Taxonomy

1. Factual Knowledge - the basic elements that students must know to be acquainted with a discipline or solve problems in it.
   - Knowledge of terminology;
   - Knowledge of specific details and elements.
2. Conceptual Knowledge - the interrelationships among the basic elements within a larger structure that enable them to function together.
   - Knowledge of classifications and categories;
   - Knowledge of principles and generalizations;
   - Knowledge of theories, models, and structures.

3. Procedural Knowledge - how to do something; methods of inquiry, and criteria for using skills, algorithms, techniques, and methods.
    - Knowledge of subject-specific skills and algorithms;
    - Knowledge of subject-specific techniques and methods;
    - Knowledge of criteria for determining when to use appropriate procedures.
4. Metacognitive Knowledge - knowledge of cognition in general as well as awareness and knowledge of one's own cognition.
    - Strategic knowledge;
    - Knowledge about cognitive tasks, including appropriate contextual and conditional knowledge;
    - Self-knowledge.

Structure of the Cognitive Process Dimension of the Revised Taxonomy

1. Remember - retrieving relevant knowledge from long-term memory.
    (a) Recognizing;
    (b) Recalling.
2. Understand - determining the meaning of instructional messages, including oral, written, and graphic communication.
    (a) Interpreting;
    (b) Exemplifying;
    (c) Classifying;
    (d) Summarizing;
    (e) Inferring;
    (f) Comparing;
    (g) Explaining.
3. Apply - carrying out or using a procedure in a given situation.
    (a) Executing;

   (b) Implementing.

4. Analyze - breaking material into its constituent parts and detecting how the parts relate to one another and to an overall structure or purpose.

   (a) Differentiating;

   (b) Organizing;

   (c) Attributing.

5. Evaluate - making judgments based on criteria and standards.

   (a) Checking;

   (b) Critiquing.

6. Create - putting elements together to form a novel, coherent whole or make an original product.

   (a) Generating;

   (b) Planning;

   (c) Producing.

Appendix B

The Relations of the Ontology of an Educational Course and Module

The following relations hold between the individuals of the ontology's classes.

1. *hasPSProfile (Programme Specification, Profile)* is used to link the individuals of the class **Programme Specification** to **Profile**. It has the following restrictions.

   (a) $\forall$ *hasPSProfile.Profile*. This restriction means, that a programme specification can have profile only chosen from the class with the same name.

   (b) $\equiv$ 1. *hasPSProfile*. This restriction means, that a programme specification must have one and only one profile.

2. *hasPSLevel (Programme Specification, Level)* is used to link the individuals of the class **Programme Specification** to **Level**. It has the following restrictions.

   (a) $\forall$ *hasPSLevel.Level*. This restriction means, that a programme specification can have level only chosen from the class with the same name.

   (b) $\equiv$. 1 *hasPSLevel*. This restriction means, that a programme specification must have one and only one level.

3. *hasRule (ProgrammeSpecification, Rule)* is used to assign the rule to a programme specification, in accordance with which the elective modules are to be chosen. It has the following restrictions.

(a) ∀ *hasRule.Rule*. This restriction means, that rules can be chosen only from the individuals of the class Rule.

(b) ≤ 1.*hasRule*. This restriction means, that a programme specification may have not more than one rule for choosing the disciplines from the group of elective modules.

4. *belongsToCategory (Competence, CompetenceCategory)* links competences to their classes. It has the following restrictions.

   (a) ∀ *belongsToCategory.CompetenceCategory*. This restriction means, that competences categories can be chosen only from the individuals of the class with the same name.

   (b) ≡ 1.*belongsToCategory*. This restriction means, that a competence should belong to one an only one category.

5. *hasCompetence (ProgrammeSpecification, Competence)* links competences to their programme specifications. It has the following restriction.

   (a) ∀ *hasCompetence.Competence*. This restriction means, that competences can be chosen only from the individuals of the class with the same name.

6. *hasMProfile (Module, Profile)* is used to link the individuals of the class **Module** to **Profile**. It has the following restrictions.

   (a) ∀ *hasMProfile.Profile*. This restriction means, that a module can have profile only chosen from the class with the same name.

   (b) ≡ 1. *hasMProfile*. This restriction means, that a module must have one and only one profile.

7. *hasMLevel (Module, Level)* is used to link the individuals of the class **Module** to **Level**. It has the following restrictions.

   (a) ∀ *hasMLevel.Level*. This restriction means, that a module can have level only chosen from the class with the same name.

   (b) ≡ 1. *hasMLevel*. This restriction means, that a module must have one and only one level.

8. *hasLearningOutcome(Module,LearningOutcome)* assign the learning outcomes to the educational modules. This relation has the following restrictions.

   (a) ∀ *hasLearningOutcome.LearningOutcome*. This restriction means, that a module can have a learning outcome only chosen from the class with the same name.

   (b) ≥ 1. *hasLearningOutcome*. This restriction means, that a module must have in the least one learning outcome.

9. *hasCourse(ProgrammeSpecification, Course)* is used to link the programme specification to all the variants of the real educational courses, which a student can take based on varying the optional and elective modules.

   It has the following restrictions.

   (a) ∀ *hasCourse.Course*. This restriction means, that a programme specification can have only the individuals of the class **Course** as the course variants.

   (b) ≥ 1 *hasCourse*. This restriction means, that in the least one variant of an educational course should be built based on a programme specification.

10. *hasCompulsoryModule(ProgrammeSpecification,Module)* is used to outline the compulsory educational modules, which a programme specification contains.

    It has an inverse property *isCompulsoryModule (Module,ProgrammeSpecification)*, which is used to receive all the programme specifications, which contain the module as compulsory.

    In the same manner the ontology contains the properties *hasOptionalModule(ProgrammeSpecification, Module)* (and its inverse *isOptionalModule(Module, ProgrammeSpecification)*), *hasElectiveModule(ProgrammeSpecification, Module)* (and its inverse *isElectiveModule(Module, ProgrammeSpecification)*), which are used to outline the optional and elective educational modules of a programme specification resprectively.

    All the direct properties (starting with "is...") have the same restriction.

    (a) ∀ *isCompulsoryModule.ProgrammeSpecification* or
    ∀ *isOptionalModule.ProgrammeSpecification* or

∀ *isElectiveModule.ProgrammeSpecification*. This restriction means, that the individuals of the class **Module** can be linked by this relation to the individuals of the class **ProgrammeSpecification** only.

Vice versae, all the inverse properties (starting with "has ...") share the following restriction.

(a) ∀ *hasCompulsoryModule.Module* or
∀ *hasOptionalModule.Module* or
∀ *hasElectiveModule.Module*. This restriction means, that compulsory, optional or elective modules of a programme specification can be chosen from the individuals of the class **Module**.

11. *hasCourseLevel(Course, Level)* is used to link the individuals of the class **Course** to **Level**. It has the following restrictions.

    (a) ∀ *hasCourseLevel.Level*. This restriction means, that a course can belong to one level only chosen from the class with the same name.

    (b) ≡ 1. *hasCourseLevel*. This restriction means, that a course must belong to a level.

12. *hasCProfile(Course, Profile)* is used to link the individuals of the class **Course** to **Profile**. It has the following restrictions.

    (a) ∀ *hasCProfile.Profile*. This restriction means, that a course can have profile only chosen from the class with the same name.

    (b) ≡ 1. *hasCProfile*. This restriction means, that a course must have one and only one profile.

13. *hasCourseLearningOutcome(Course,LearningOutcome)* links the course and the learning outcomes of the modules, which it consists of.

    (a) ∀ *hasLearningOutcome.LearningOutcome*. This restriction means, that a course can have a learning outcome only chosen from the class with the same name.

    (b) ≥ 1. *hasLearningOutcome*. This restriction means, that a course must have in the least one learning outcome.

14. *hasModule (Course,Module)* enables inclusion of the educational modules into the educational courses. When talking about a particular educational course, we do not further need to distinguish modules by their type (compulsory, optional, elective), as at this point we suppose, that a student will take or had already studied all of them. However, if it is necessary it is possible to infer the type of each module's inclusion based on the relations between a programme specification and a course, and the modules. The following restrictions apply to this property.

    (a) $\forall$ *hasModule.Module*. This restriction means, that a course can have only the individuals of the class **Module** as the modules.

    (b) $\geq$ 1. *hasModule*. This restriction means, that an educational course should contain in the least one module.

Appendix C

The Relations of the Data Domain Sub-Ontology

The following relations hold between the individuals of the ontology's classes.

1. *belongsToTopic (Keyword, Topic)* connects the keywords to the topics, which they belong to. The inverse to it is the property *hasKeyword (Topic, Keyword)*, which enables retrieval of all of the keywords of a particular topic. The following restriction applies to this property.

   (a) $\forall$ *belongsToTopic.Topic*. This restriction means, that a keyword may be linked by this property to an individual of the class **Topic** only.

2. *containsNounDD (Keyword, NounDD)* links the nouns to the collocations, which contain them. This property has the following restrictions.

   (a) $\forall$ *containsNounDD.NounDD*. This restriction means, that a keyword may be linked by this property to an individual of the class **NounDD** only.

3. *containsAdjDD (Keyword, AdjectiveDD)* outlines the adjective, which the collocation or a keyword contains. It has the following restrictions.

   (a) $\forall$ *containsAdjDD.AdjectiveDD*. This restriction means, that a keyword may be linked by this property to an individual of the class **AdjectiveDD** only.

(b) $\leq$ 1. *containsAdjDD* restricts the possible number of adjectives in a collocation to zero or one.

4. *hasRelation (Keyword, Keyword)* indicates relations between the keywords. In this work we distinguish between the three main types of relations, which can hold between the concepts. The type is defined by a preposition, construction or a verb, used between the keywords in a collocation. The prepositions, verbs and constructions are included into the ontology as subproperties of the relations of the corresponding types.

   (a) *hasPrepositionalRelation (Keyword, Keyword)* is the relation, which is defined in the text by prepositions "of", "from" and others, for example, data from database. Here the keywords "data" and "database" are connected by the preposition "from". We called this type "prepositional relation".

   (b) *hasDetalisationRelation (Keyword, Keyword)* is the relation, defined by constructions "such as", "for example" and others. For instance, "database management system, such as Oracle". The keywords "database management system" and "Oracle" are linked by the construction "such as", which means that Oracle is a type of database management system. We called this type "detalisation relation".

   (c) *hasVerbalRelation (Keyword, Keyword)* is the relation, identified by the verb used between the keywords. For example, "database implements a data model". The keywords "database" and "data model" are linked by the verb "implement".

Appendix D

The Relations of the Sub-Ontology of the Learning Outcomes

The ontology of a learning outcome contains the following relations between the individuals of its classes.

1. *hasActionVerb (LearningOutcome, ActionVerb)* links the action verbs, which form an action verb construction, to a particular learning outcome. This property has the following restrictions.

   (a) *hasActionVerb.ActionVerb* means, that an action verb can be chosen only from the class with the same name.

   (b) $\geq$ 1. *hasActionVerb* means, that any learning outcome must have in the least one action verb.

2. *hasDDOConstruction (LearningOutcome, DDOConstruction)* links the identifiers of the data domain object constructions to the learning outcomes. This property has the following restrictions.

   (a) *hasDDOConstruction.DDOConstruction* shows, that the identifiers can be chosen from the class DDOConstruction only.

   (b) $\equiv$ 1. *hasDDOConstruction* means, that any learning outcome must have one and only one data domain object construction.

3. *hasDataDomainObject (DDOConstruction,DataDomainObject)* enables connection of the data domain objects into one construction. This property has the following restrictions.

   (a) *hasDataDomainObject*.*DataDomainObject* means, that the data domain object can be chosen only from the class with the same name.

   (b) $\geq$ 1. *hasDataDomainObject* means, that any data domain object construction must have in the least one data domain object.

4. *hasComplexity (DataDomainObject, Complexity)*, *hasImportance (DataDomainObject, Importance)*, *hasSize (DataDomainObject,Size)* enables characterisation of the data domain objects in the terms of complexity, importance and size. Each of this properties can have only the individuals of the classes **Complexity**, **Importance** or **Size** as their range respectively.

5. *hasSClause (LearningOutcome,SClause)* links the specifying clause to the main part of a learning outcome. The restriction *hasSClause*.*SClause* means that specifying clauses' identifiers can be chosen only from the class **SClause**.

6. *hasSC (SClause,SC)* assigns particular subordinate constructions to their specifying clauses. Again the constructions can be chosen only from the class SC (*hasSC*.*SC*).

7. *hasSCDDOConstruction (SClause, DDOConstruction)* is the analogue of the property *hasDDOConstruction* as applied to the specifying clause instead of main part of a learning outcome. It links the identifiers of the data domain object constructions to the specifying clauses.

8. *hasSCActionVerb (SClause, ActionVerb)* in the same manner as the previous property, is the analogue of the property *hasActionVerb* for the main part of a learning outcome.

Appendix E

The Parts of Speech and Other Symbols

The following parts of speech, punctuation marks and other symbols are utlised by the grammars. All the definitions of the parts of speeches were adopted from [1] (last accessed on May, 5, 2013).

Parts of Speech

Predeterminer (PDT) is "a word that occurs before a determiner, typically quantifying the noun phrase". It is used to mark the elements preceding articles in the noun phrases.

Example: all, quite.

Determiner (DT) is "a modifying word that determines the kind of reference a noun or noun group has". It is used to mark articles.

Example: a, an, the.

Preposition (Prep) is "a word governing, and usually preceding, a noun or pronoun and expressing a relation to another word or element in the clause".

Example: in, for, from.

Possessive Pronoun (PossPr) is "a pronoun indicating possession".

Example: your, its, their.

Noun is a word "used to identify any of a class of people, places, or things (common noun), or to name a particular one of these (proper noun)". In this work we also treat abbreviations in the same way with nouns.

Example: database, model, knowledge.

Adjective (Adj) is "a word naming attribute of a noun".

Example: thorough, small, relational.

Verb is "a word used to describe an action, state or occurrence, and forming the main part of the predicate of a sentence". In the Grammar of the Learning Outcomes we address as **Verb** the verbs in base, present and past tenses, including past , but excluding present participle.

Example: demonstrate, create, evaluate, produce, know, implement.

Modal Verb (MV) a special type of verbs in the English language, which, according to [1], "expresses necessity or possibility".

Example: must, shall, will, should, would, ought to, can, could, may, might.

Present Participle (PP) is "the form of a verb, ending in *-ing* in English, which is used in forming continuous tenses, [. . .] alone in non-finite clauses, [. . .] as a noun, [. . .] and as an adjective".

Example: adding, interrogating, caching.

Past Participle (PastP) is "the form of a verb, typically ending in *ed* in English, which is used in forming perfect and passive tenses and sometimes as an adjective".

Example: stored, organised, object-oriented.

Adverb (Adv) is "a word or phrase that modifies the meaning of an adjective, verb or other adverb, expressing manner, place, time, or degree".

Example: critically, thoroughly.

Punctuation and Other Symbols

Apart from the parts of speeches, the both grammars share the following non-terminal symbols.

1. **Abbreviation (Abbr)** is ”a shortened form of a word or a phrase” [1], which usually consists of the first letters of the word sequence, written in the upper case.

   Example: SQL, API, JDK.

2. *CC* $\rightarrow$ *and* | *but* | *or*

   CC stands for a coordinating conjunction, which is a ”conjunction placed between words, phrases, clauses, or sentences of equal rank”.

3. *Sep* $\rightarrow$; | .

   Sep stands for punctuation marks semi-colon (”;”) or full-stop (”.”). We call them ”separators”, because they are used for separating either keywords or learning outcomes from each other.

4. The non terminal symbols **LP** and **RP** are used to mark opening and closing parentheses correspondingly. We use special non-terminal in order not to mix them up with operator ”brackets”.

Appendix F

The Examples of Short Course and Module Templates

The short course template (SCT) is a document, which should be prepared based on a programme specification. It should contain the title, course code, level, faculty, department, university, the number of credits and the list of its modules (possibly module codes).

Example of an SCT.

Course Title: Information Technology

Course Code: CC088T

Level : Postgraduate

Faculty: Technology

Department: Computer Science

University: DMU

Modules: COMP5101; COMP5102; COMP5103; COMP5104; COMP5208.

The short module template (SMT) must contain the basic information about the module, including the title, code and number of credits, the section with its keywords and another one with the learning outcomes. The keyword section must

have a title ”Keywords”. It should contain the topics and their keywords. The topic should precede the list of its keywords and be separated from them by a colon. The section ”Learning outcomes” should contain the list of its learning outcomes.

Example of an SMT.

Module Title: Database Design Concepts

Module Code: COMP5100

Level : Postgraduate

Profile: Computer Science

Credits: 15

Keywords:

1. Requirements Analysis: use case diagram; relational data modelling, first level data design, conversion to tables; data flow diagram.
2. Implementation of databases: mapping to a database schema using Data Definition Language; database querying using Data Manipulation Language; validation of input data; verification of input data.
3. The business environment: organisational structures; basic business model; IT in organisations; risk assessment.

Learning outcomes:

1. Demonstrate an understanding of business environments and the use of databases within those environments.
2. Produce a database design based on a given data model.
3. Map the design to a relational database management system such as Access and produce non-trivial queries to meet user requirements, using SQL.

Appendix G

The Phases of the Keywords' and Learning Outcomes' Annotators

The Phases of the Keywords' Annotator

Phase 1. POS-tags Gathering Phase 1 receives as input the tokens and lookups, produced by the previous processing resources. The control type is set to "brill". All the rules have equal priority. The range of Hepple-tags, assigned to the words by the POS-tagger is richer, then the Grammar of a Keyword needs. Thus the tags are grouped and associated with the sets of GK's non-terminals. At this stage we gather the entities "PreKW" (rule 3), "NounDD" (rule 5a.1), "Verb", "PastP" (Past Participle), "MV" (modal verb),"Adverb", "AdjDD" (rule 4), "PP" (Present Participle) and "Prep" (Preposition) based on the Hepple tags. The correspondence between the groups of tags, non-terminals and the grammar's rules is presented in the table G.1. Additionally, the abbreviations are annotated with tag "Abbrev" (rule 5a.2). We mark as abbreviation a token, which consists of the capital letters only. The punctuation symbols and conjunctions are marked "CC" and "Sep", the parentheses are annotated "LP" and "RP".

Example:

Keywords:

[NounDD Implementation] [Prep of] [NounDD databases] [Sep :] [NounDD (according to rule 5b) mapping] to [DT a] [NounDD database] [NounDD

Table G.1: The Correspondence between the Groups of Hepple tags and the Non-terminals in the GK and GLO

Non-terminal Symbol	**Set of Hepple-Tags/String**
PDT	PDT
DT	DT
PossPr	PRP, PRP, *PRPR*
NounDD	NN, NP, NPS, NNP, NNS, NNPS
Verb	VB, VBD, VBN, VBP, VBZ
PastP	VBN
MV	MD
Adverb	RB
AdjDD	JJ
PP	VBG
Preposition	IN
CC	″and″, ″or″, ″but″
Sep	″;″,″.″
LP	″(″
RP	″)″

schema] [PP using] [NounDD Data] [NounDD Definition] [NounDD Language] [LP (] [Abbr DDL] [RP)] [Sep ;] [NounDD database] [NounDD (according to rule 5c) querying] [PP using] [NounDD Data] [NounDD Manipulation] [NounDD Language] [AdjDD such] [Prep as] [Abbr SQL] [Sep ;] [NounDD validation] [Prep of] [NounDD input] [NounDD data][Sep ;] [NounDD verification] [Prep of] [NounDD input] [NounDD data] [Sep .]

Learning Outcomes:

[Verb Discuss] [NounDD Database] [NounDD Management] [NounDD Systems], [AdjDD such] [Prep as] [NounDD Oracle] [Sep .]

Phase 2. Keyword Annotation This phase is devoted to looking for the keywords in the text (″Keyword″). The phase receives the annotations of type ″Token″, ″PreKW″, ″AdjDD″, ″NounDD″, ″PastP″, ″PP″, ″CC″ and ″Sep″. The control style is set to ″appelt″, because we used two groups of rules with different priorities. The statements 5b and 5c are included directly into the JAPE-rules for recognition of Keyword.

Example:

Keywords:

[Keyword Implementation] of [Keyword databases]: [Keyword mapping] to a [Keyword database schema] using [Keyword Data Definition Language] ([Keyword DDL]); [Keyword database querying] using [Keyword Data Manipulation Language] such as [Keyword SQL]; [Keyword validation] of [Keyword input data]; [Keyword verification] of [Keyword input data].

Learning Outcomes:

Discuss [Keyword Database Management Systems], such as [Keyword Oracle].

Phase 3. Abbreviation Phrases' Annotation This phase annotates the abbreviation phrases ("AbbrPhrase", rule 6). It receives as input the following non-terminals: "RP", "LP", "Abbr" and "Keyword. The control style is set to "brill", as the phase contains only one rule.

Example:

Keywords:

Implementation of databases: mapping to a database schema using [AbbrPhrase Data Definition Language (DDL)]; database querying using Data Manipulation Language such as SQL; validation of input data; verification of input data.

Learning Outcomes:

Discuss Database Management Systems, such as Oracle.

Phase 4. Annotating the Relations The fourth phase is used for annotation of the relations ("PrepRel", "VerbRel", "DetRel", rule 7). The input non-terminals are "PreKW", "AdjDD", "NounDD", "Abbr", "AbbrPhrase", "Verb", "PP", "Keyword" and "Prep". The control style is set to "appelt".

Example:

Keywords:

Implementation [PrepRel of] databases: mapping [PrepRel to] a database schema [VerbRel using] Data Definition Language (DDL); database querying [VerbRel using] Data Manipulation Language [DetRel such as] SQL; validation [PrepRel of] input data; verification [PrepRel of] input data.

Learning Outcomes:

Discuss Database Management Systems, [DetRel such as] Oracle.

Phase 5. Annotating the Relations' Phrases The fifth phase marks the relation phrases (”PrepRelPhrase”, ”DetRelPhrase”, ”VerbRelPhrase”, rule 8). The inputs are ”Token”, ”Keyword”, ”PrepRel”, ”VerbRel”, ”DetRel”, ”Abbr”, ”AbbrPhrase” and ”CC”. The control style is set to ”appelt”.

Example:

Keywords:

[PrepRelPhrase Implementation of databases]: [PrepRelPhrase mapping to a ”VerbRelPhrase database schema] using Data Definition Language (DDL)”; ”VerbRelPhrase database querying using [DetRelPhrase Data Manipulation Language” such as SQL]; [PrepRelPhrase validation of input data]; [PrepRelPhrase verification of input data].

Learning Outcomes:

Discuss Database Management Systems, [DetRelPhrase such as Oracle].

Phase 6. Annotating the Topics The last phase annotates the topics (”Topic”, rule 1). The input non-terminals contain ”Token”, ”Keyword”, ”AbbrPhrase”, ”PrepRelPhrase”, ”DetRelPhrase”, ”VerbRelPhrase”.

Example:

Keywords:

[Topic Implementation of databases]: mapping to a database schema using Data Definition Language (DDL); database querying using Data Manipulation Language such as SQL; validation of input data; verification of input data.

Learning Outcomes:

Discuss Database Management Systems, such as Oracle.

The Phases of the Learning Outcomes' Annotator

Phase 1. POS-tags Gathering

The first phase of the GLO is alike with the same phase and implementation of the Grammar of the Keywords with respect to matching of the Hepple tags to non-terminals.

Phase 1 receives as input the tokens and lookups, produced by the previous processing resources. The control type is set to "brill". All the rules have equal priority. At this stage we find the non-terminals "NounDD" (Group 3, rule 6), "NounCN" (Group 2, rule 7), "Verb", "PastP" (Past Participle), "MV" (modal verb),"Adverb", "AdjDD" (Group 3, rule 4), "PP" (Present Participle) and "Prep" (Preposition) based on the Hepple tags. The correspondence between the groups of tags, non-terminals and the grammar's rules is presented above in the Table G.1. Additionally, the abbreviations are annotated with tag "Abbrev" (rule 5a.2). We mark as abbreviation a token, which consists of the capital letters only. The punctuation symbols and conjunctions are marked "CC" and "Sep", the parentheses are annotated "LP" and "RP".

Phase 2. Markup of the Action Verbs and Other Ontological Entities

Phase 2 receives as input the tokens, lookups, annotations of the types "Adverb", "PP" and "Verb", "NounCN" and "Prep" produced by the previous processing resources. The control type is set to "appelt". The rules are divided into groups according to their priority.

The rules in this phase have three priority levels. The highest value is given to those, annotating the text based on ontology entities. They include markup of the words (or word sequences) with the annotations "SC" (Group 4, rule 2), if they exist in the ontology as individuals of the corresponding class, and "Char", if they can are instances of one of the classes "Size", "Complexity", "Importance" (Group 3, rules 8 to 10). Another rule with the highest priority is used to assign the "AV" annotations to the sequences, which appear in the ontology as instances of the classes "ClusterAV", "SubClusterAV" and "DataDomainActionVerb" (Group 2, rule 6).

The middle priority is given to the rule, which attempts to find the action verbs, which are not present in the provided ontology of D. Krathwohl's model. In order to do this, we apply the rule, which finds the sequences in accordance with the rules 2 and 3 from the Group 2 of the Grammar of the Learning Outcomes.

The lowest priority is given to the rule, which identifies "AVAdd" (Group 2, rule 5).

Phase 3. Markup of the Keywords

Phase 3 is aimed at finding the keywords in the sense of the learning outcomes. It takes the non-terminals "Token", "Lookup", "NounDD", "AV", "SC", "Char", "PossPr", "PastP", "Abbrev" and "AdjDD" annotations as input. The control type is set to "appelt". The rule implements the rule 3 from the Group 3 of the Grammar of the Learning Outcomes. The additional input annotations "AV" and "SC" are used in the rules to negate double annotations, where they may appear. For example, we consider annotation "NounDD" as data domain noun only if it is not subsumed by "AV" or "SC" annotation. The keywords inferred at this phase are marked "Keyword2".

Phase 4. Markup of the Data Domain Objects and the Action Verb Constructions

The fourth phase is aimed at gathering data domain objects from the keywords, action verb and additional action verb constructions in accordance with the rule 2, Group 3 and the rules 1 and 4, Group 2 of the Grammar of the Learning Outcomescorrespondingly. It takes "Token", "Keyword2", "Characteristic", "Preposition", "CC" and "ActionVerb" annotations as input. The control style is "appelt".

The phase contains two groups of rules with different priorities: firstly, the action verb and additional action verb constructions are gathered, afterwards the data domain objects are annotated in accordance with the rules of the Grammar of the Learning Outcomes.

Phase 5. Markup of the Data Domain Object Constructions

This phase takes "DDO", "Prep" and "CC" annotations as input, the control style is "appelt". It implements the rule 1, Group 3 from the Grammar of the Learning Outcomesand markups the data domain object constructions with "DDOC" annotations.

Phase 6. Markup of the Main and Specifying Clauses

This phase receives ″Token″, ″SC″, ″DDOC″, ″AVC″, ″AVCAdd″, ″LP″, ″RP″, ″PP″ and ″CC″ annotations as input. The control style is appelt.

At the phase 6 we recognise the main clause of the learning outcome ″MainLO″ (rule 3 from Group 1) and the specifying clauses ″SClause″ (rule 1 from Group 4). The higher priority is assigned to the rule, recognising the specifying clauses to avoid double annotation of the parts of texts in the cases, when a constituent of the SClause has the same structure as a whole MainLO.

Phase 7. Markup of the Learning Outcomes

This phase takes the ″MainLO″ and ″SClause″ annotations as input and contains only one rule (the control style is ″appelt″).

Phase 7 is used to annotate the whole learning outcomes in accordance with the rule 2 from Group 1. This is enough to populate the ontology with the learning outcomes and we do not need to fire the rule 1 from the Group 1, as it does not matter for the ontology population, how the LOs are distributed among the natural language sentences NLLOSentence.

Appendix H

The Illustration of the Implementation of the Algorithms

```
//Calculate similarity between adjectives if and only if both keywords contain AdjDD
                //if both keywords contain AdjDDs
     {//compute structural and semantic similarity for the adjectives
simAdjDD = CalculateAdjectiveToAdjectiveSimilarity(KW1.AdjDD,KW2.AdjDD);}

//we calculate the weight of similarities between adjectives dividing 1 by the  number of
words contained in the longer Keyword
                 double weightAdjDD=0;
                 if both keywords contain NounDDs
                { weightAdjDD= 1.0/(Math.max (KW1.NounDD.size(), KW2.NounDD.size())+1);
                  }

                 else if only the first keyword contains NounDD
                 {weightAdjDD=1.0/(KW1.NounDD.size()+1);
                 }

                 else if only the second keyword contains NounDD
                 {weightAdjDD=1.0/(KW2.NounDD.size()+1);
                 }

                 else if none of the keywords contains NounDD
                 {weightAdjDD=1.0;
                 }

          //calculate the weight of the similarity between NounDDs contained in the Keywords
                 double weightNDD = 1-weightAdjDD;
 //if both keywords contain NounDDs, calculate the similarity between the sets of the
NounDDs
int rowNDD=0;
while (ItKW1NounDD.hasNext())
                 {
                  int columnNDD=0;
               ...
                    while (ItKW2NounDD.hasNext())
                    {
                         …
//compute structural and semantic similarity for the nouns
         double simNDDtmp = CalculateNounToNounSimilarity (KW1NDD,KW2NDD);
                 resultsNDD [rowNDD] [columnNDD] = simNDDtmp;
                       columnNDD++;
                  }
                 rowNDD++;
                }
     ResultMatrix mrxNDD = new ResultMatrix();
// the similarity between the sets is calculated in accordance with greedy strategy
      simNDD = mrxNDD.CalculateGreedyIterative(resultsNDD);
// the strutural and semantic similarity between two keywods
double AdjAndNounDDSim = weightAdjDD*simAdjDD+weightNDD*simNDD;}
```

Figure H.1: The Implementation of the Structural and Semantic Similarity between Keywords

```
        // the strutural and semantic similarity between two keywods
double AdjAndNounDDSim = weightAdjDD*simAdjDD+weightNDD*simNDD;

//if both keywords contain detalisation relations
{ //compute the similarity between the ranges of all these relations
double DetRelRange= CalculateKeywordPartWholeRelationRangeSimilarity(KW1,KW2);
          if (DetRelRange!=0.0)
                    {//compute the total similarity between the detalisation relations
                    simDetRel = 0.5* (AdjAndNounDDSim+ DetRelRange);
                    }
                    else simDetRel =0.0;
              }

              if both keywords contain prepositional relations
{//compute the similarity between the ranges of all these relations
double PrepRelRange = CalculateKeywordStructRelationRangeSimilarity(KW1,KW2);
                    if (PrepRelRange!=0.0)
                    {//compute the total similarity between the prepositional relations
                    simPrepRel =      0.5 * (AdjAndNounDDSim+PrepRelRange);
                    }
                    else simPrepRel =0.0;
                    }

              if both keywords contain verbal relations
{ //compute the similarity between the verbs and ranges of all these relations
double VerbRangeRel = CalculateKeywordVerbalRelationAndRangeSimilarity(KW1,KW2);
                     if (VerbRelRange!=0.0)
                    {//compute the total similarity between the verbal relations
                     SimVerbRel = 0.5 * (AdjAndNounDDSim+VerbRangeRel);
                     }
                    else simVerbRel =0.0;
                    }
double denominator =0.0;
if one type of relations was compared denominator =1;
If two types of relations were compared denominator = 2;
If three types of relations were compared denominator = 3;

//compute the total similarity between all relations of the keyword
if (denominator !=0)
    { double simAllRel = (simPartWholeRel+simStructRel+simVerbRel)/denominator; }
```

Figure H.2: The Implementation of the Relation Similarity between Keywords

```
//Compute similarity between the data domain object constructions
DDOCsimilarity = new DataDomainObjectSimilarity(LO1.DDO, LO2.DDO);

  // if the similarity between the data domain object constructions >=threshold
                    {
                    \\compute the similarity between the action verb constructions
                   AVsim = new AVSim( LO1.AV, LO2.AV,"MainLO");

                   \\compute the similarity between the main clauses of the learning outcomes
                   MainLOsim = (DDOCsimilarity+AVsim.AllAVsim)/2;

                   \\similarity between the LOs equals the similarity between their main clauses
                    simLO = MainLOsim

          // if both outcomes contain specifying clauses
                   {
                        int rowSCl=0;
                     ...
                        while (ItLO1SCl.hasNext())

                        {       ...
                              int columnSCl=0;
                           while (ItLO2SCl.hasNext())
                   //if the specifying clauses are comparable, compare them

                              {  ...
                                 SClauseSim SClSim = new SClauseSim(SCl1 , SCl2);
                                    }
                                  // else set similarity to zero

                                    { SClsim=0; }
                                      resultsSCl [rowSCl] [columnSCl] = Sclsim;
                                        columnSCl++;
                              }
                              rowSCl++;
                           }

                           ResultMatrix mrx = new ResultMatrix();
          //aggregate the results in accordance with the greedy iterative strategy
                           double SClsimavg = mrx.CalculateGreedyIterative(resultsSCl);
//similarity between the LOs equals simple average between their main
//and specifying clauses' similarities
                           simLO = 0.5*MainLOsim + 0.5*SClsimavg;}
```

Figure H.3: The Implementation of Learning Outcomes' Alignment

Appendix I

The Questionnaire

Below we represent the example of a questionnaire, which the respondents received. Firstly, they were asked a few preliminary questions to identify their area of expertise. Afterwards they were asked to evaluate the short module templates pairwise.

QUESTIONNAIRE

The main aim of the questionnaire is to evaluate the similarities between the module marked as "Master" against other module templates (MTs) in the**Package 1** and **Package 2**. **The similarity between the Modules is the extent to which their contents (keywords, learning outcomes) overlap. This means that the similarity is independent on the side from which you start the comparison.**

Could you please compare the Master MT against each of the other templates based on information provided in the corresponding sections of the MTs and fill in the questionnaire?
In order to evaluate the similarity between the sections of the Master module and another module from the package, please choose the most appropriate similarity value of your choice (questions 5a, 6a, 7a, 8a, 9a). Could you please also provide the level of your confidence about each answer (questions 5b, 6b, 7b, 8b, 9b)?

Preliminary Data

1. What is your current employment status? You can choose more than one answer if applicable.

- ☐ PhD student
- ☐ Work in academia
- ☐ Work in business
- ☐ Other:

2. What is your field of expertise? You can choose more than one answer if applicable.

- ☐ Information technologies
- ☐ Other:

3. Do you have any experience in teaching (e.g. reading lectures, supervising students etc.)?

- ☐ Yes
- ☐ No

4. Have you ever designed an educational Module or Course?

- ☐ Yes
- ☐ No

PACKAGE 1

COMPARISON 1

Master Module 1. Data Analysis and Database Design	Module 1.1 Database Design Concepts
Keywords	
1. *First level data design*: top-down data modelling, entity-relationship diagrams, primary keys, foreign keys, decomposition of m:m relationships, connection traps, recursive relationships; bottom-up data modelling, normalization, determinancy diagrams. 2. *Second level data design*: flexing the first level data design to meet performance requirements, implementation of the data model using a RDBMS such as Oracle. 3. *Data analysis*: use of SQL to retrieve data from large database; presentation of data. 4. *Database Management Systems (DBMS) issues*: entity integrity, referential integrity; transaction management, concurrency, recovery; database access control; database security.	1. *Requirements Analysis*: use case diagram; relational data modelling, first level data design, conversion to tables; data flow diagram. 2. *Implementation of databases:* mapping to a database schema using Data Definition Language; database querying using Data Manipulation Language; validation of input data; verification of input data. 3. *The business environment*: organisational structures; basic business model; IT in organisations; risk assessment.
Learning Outcomes	
1. Produce a data model that accurately reflects a moderately complex business scenario. 2. Produce a database design based on a given data model and map the design to a Relational Database Management System such as Oracle. 3. Perform data analysis on large data sets to solve a given problem. 4. Discuss key database management systems issues. 5. Use SQL to retrieve data from large database. 6. Discuss Database Management Systems, such as Oracle.	1. Demonstrate an understanding of business environments and the use of databases within those environments. 2. Produce a database design based on a given data model. 3. Map the design to a relational database management system such as Access and produce non-trivial queries to meet user requirements, using SQL. 4. Discuss key database technologies, management systems issues and database environments. 5. Implement and test a database design 6. Implement a small database application for a given scenario. 7. Create, manipulate, and interrogate

	database tables using SQL. 8. Create a range of standard SQL scripts to create and manipulate tables, adding and interrogating data.

5a. The Master Module 1 and Module 1.1 are similar.

Answer / Criteria	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Keywords					
Learning Outcomes					
Overall					

5 b. Please provide your level of confidence in the answer to question 5a

Low	Moderate	High

COMPARISON 2

Master Module 1. Data Analysis and Database Design	Module 1.2 Data Models
Keywords	
1. *First level data design*: top-down data modelling, entity-relationship diagrams, primary keys, foreign keys, decomposition of m:m relationships, connection traps, recursive relationships; bottom-up data modelling, normalization, determinancy diagrams. 2. *Second level data design*: flexing the first level data design to meet performance requirements, implementation of the data model using a RDBMS such as Oracle. 3. *Data analysis*: use of SQL to retrieve data from large database; presentation of data. 4. *Database Management Systems (DBMS) issues*: entity integrity, referential integrity; transaction management, concurrency, recovery; database access control; database security.	1. *The key concepts:* data domain of the database, information, data, data models, entity-relationship data model, the methodology for synthesis of data models of the data domain 2. *The data models:* hierarchical data model, oriented graphs, recursive creation of hierarchies; network data model; relational data model, relational schemes, relational database design, optimization of queries,relational calculus for tuple variables and variable domains, functional and multivalued dependencies, decomposition schemes normalization of relational data models. 3. *Database Management Systems (DBMS):* DBMS Access; description languages and data manipulation; language QBE; language SQL; application development; CASE tools. 4. *The models of knowledge bases*: intelligent information retrieval systems, hypertext systems, computational logic systems, expert systems; overview of the main logical-linguistic models of knowledge bases: semantic networks, frames, production system.
Learning Outcomes	
1. Produce a data model that accurately reflects a moderately	1. Know formulas and theorems describing information processing in databases.

complex business scenario. 2. Produce a database design based on a given data model and map the design to a Relational Database Management System such as Oracle. 3. Perform data analysis on large data sets to solve a given problem. 4. Discuss key database management systems issues. 5. Use SQL to retrieve data from large database. 6. Discuss Database Management Systems, such as Oracle.	2. Know and apply procedures for the synthesis of information structure model of data domains. 3. Know and discuss key data models of databases and knowledge bases. 4. Know and apply methods of synthesis and optimization of database structures, methods of optimization of the database queries. 5. Understand and apply the systematic study of design decisions on the structure of data models and databases, the architecture of a data bank and its components. 6. Discuss and develop data models at the stage of engineering design.

6a. The Master Module 1 and Module 1.2 are similar.

Answer / *Criteria*	*Strongly Agree*	*Agree*	*Neutral*	*Disagree*	*Strongly Disagree*
Keywords					
Learning Outcomes					
Overall					

6 b. Please provide your level of confidence in the answer to question 6a

Low	*Moderate*	*High*

COMPARISON 3

Master Module 1. Data Analysis and Database Design	Module 1.3 Data Models
Keywords	
1. *First level data design*: top-down data modelling, entity-relationship diagrams, primary keys, foreign keys, decomposition of m:m relationships, connection traps, recursive relationships; bottom-up data modelling, normalization, determinancy diagrams. 2. *Second level data design*: flexing the first level data design to meet performance requirements, implementation of the data model using a RDBMS such as Oracle. 3. *Data analysis*: use of SQL to retrieve data from large database; presentation of data. 4. *Database Management Systems (DBMS) issues*: entity integrity, referential integrity;	1. *Architecture of a database:* the main components of a local database; the architecture of a data bank based on: client-server technology, the model file server, the model of access to remote data, the model of the database server, the model application server. 2. *Relational Database Management Systems (DBMS):* the main functions and components of data management systems, data description language (DDL), data manipulation language (DML), query language (QL), query optimizer, data dictionary; Object-oriented DBMS: system

transaction management, concurrency, recovery; database access control; database security.	classes, user interfaces, query tool, clustering and caching; Distributed Databases: levels of data distribution and processing, distributed queries and distributed transactions, managing parallel execution of transactions in a distributed environment; Internet - technologies and databases. 3. *Structured Query Language (SQL)*: commands for data description, data manipulation, queries, additional commands for data management, SQL functions; Procedural SQL: triggers, stored procedures, stored functions, PL / SQL. 4. *Data structures in computer memory*: list structures, tree and network structures; methods of organizing file systems: index structures, hash files, inverted structures. 5. *Methods for special treatment*: ensuring data integrity, securing your data in the database; optimization of queries; organization of parallel data processing; transactions.
Learning Outcomes	
1. Produce a data model that accurately reflects a moderately complex business scenario. 2. Produce a database design based on a given data model and map the design to a Relational Database Management System such as Oracle. 3. Perform data analysis on large data sets to solve a given problem. 4. Discuss key database management systems issues. 5. Use SQL to retrieve data from large database. 6. Discuss Database Management Systems, such as Oracle.	1. Know formulae and theorems describing informati on processing in the databases. 2. Know and use quantities that characterize the quality of information processing in databases and methods of optimization of data processing. 3. Know and apply methods of synthesis and optimization of database structures, methods of optimization of the database queries. 4. Know and understand architectures of databases. 5. Understand and apply the systematic study of design decisions on the structure of databases, the architecture of a

	database and its components. 6. Produce a database design based on a given data model. 7. Create SQL scripts for data retrieval and modification. 8. Create PL/SQL stored procedures. 9. Understand and discuss data structures in computer memory. 10. Know and discuss key methods for special treatment of databases.

7a. The Master Module 1 and Module 1.3 are similar.

Answer / *Criteria*	*Strongly Agree*	*Agree*	*Neutral*	*Disagree*	*Strongly Disagree*
Keywords					
Learning Outcomes					
Overall					

7 b. Please provide your level of confidence in the answer to question 7a

Low	*Moderate*	*High*

PACKAGE 2

COMPARISON 4

Master Module 2. Object-Oriented Software Design and Development	Module 2.1 Object-Oriented Systems Analysis and Design
Keywords	
1. *Usage of the Java2 SDK API*: Java collection classes; GUI & event handling; exceptions; file handling; persistence; collections; utilities such as: dates, formatting, tokenizing. 2. *UML notation*: class diagrams; collaboration diagrams; use cases; object-oriented analysis; program design. 3. *Implementation of associations*: composition and aggregation, user defined classes; data representation; interfaces; abstraction. 4. *Application of software architectures*: design patterns; Model-View-Controller software architecture; composite; command; decorator. 5. *Quality issues in software development*: documentation for maintainability and reusabilty, use of 'javadoc' tool; unit and system testing; correctness; robustness; HCI issues.	1. *Systems Analysis:* the object-oriented paradigm; RAD approach to systems development; techniques for analysis and modelling of logical requirements; UML models; project lifecycle. 2. *Systems Design*: object design, assigning responsibilities, constraints and integrity; software architecture, layering and partitioning, packaging subsystems, deployment; interface design. 3. *UML modelling techniques*: Use Cases, Activity Diagrams, Class Diagrams, Interaction Diagrams, Statecharts, Package, Component and Deployment Diagrams. 4. *Systems development methodology*: Unified Software Development Process.

Learning Outcomes	
1. Use Java to implement standard object-oriented designs given in UML. 2. Use UML to document object-oriented designs. 3. Address issues of quality, maintainability, correctness, and robustness with respect to software design and development. 4. Make effective use the Java Software Development Kit Application Programming Interfaces.	1. Explain the object-oriented paradigm and identify the significant issues that are addressed when developing object-oriented analysis and design models; 2. Construct a set of analysis and design models using an appropriate range of object modelling techniques that represent different perspectives of a particular systems development (e.g. Build uml models such as use cases, class diagrams, activity diagrams, interaction diagrams, state charts and package diagrams); 3. Use a case tool to build and document analysis and design models in a consistent manner. 4. Describe how analysis and design activities are organized within a leading systems development methodology (e.g. Unified software development process) and discuss the contribution of the methodology in the production of quality software systems.

8a. The Master Module 2 and Module 2.1 are similar.

Answer / *Criteria*	*Strongly Agree*	*Agree*	*Neutral*	*Disagree*	*Strongly Disagree*
Keywords					
Learning Outcomes					
Overall					

8 b. Please provide your level of confidence in the answer to question 8a

Low	*Moderate*	*High*

COMPARISON 5

Master Module 2. Object-Oriented Software Design and Development	Module 2.2 Introductory Java Programming
Keywords	
1. *Usage of the Java2 SDK API*: Java collection classes; GUI & event handling; exceptions; file handling; persistence; collections; utilities such as: dates, formatting, tokenizing. 2. *UML notation*: class diagrams; collaboration diagrams; use cases; object-oriented analysis; program design.	1. *Structured Programming*: development environments; source code; object code; compilation; data types; variables; assignments; arithmetic operators; expressions; selection; iteration; arrays; methods/procedures; parameter passing. 2. *Basics of Object-Oriented Programming*:

3. *Implementation of associations*: composition and aggregation, user defined classes; data representation; interfaces; abstraction. 4. *Application of software architectures*: design patterns; Model-View-Controller software architecture; composite; command; decorator. 5. *Quality issues in software development*: documentation for maintainability and reusabilty, use of 'javadoc' tool; unit and system testing; correctness; robustness; HCI issues.	classes; objects; attributes and methods; encapsulation. 3. *Implementing simple classes in Java*: UML diagrams; simple inheritance; interfaces; the String class; Java SDK; case studies; simple algorithms; software quality; testing; maintenance; reliability; quality; documentation.
Learning Outcomes	
1. Use Java to implement standard object-oriented designs given in UML. 2. Use UML to document object-oriented designs. 3. Address issues of quality, maintainability, correctness, and robustness with respect to software design and development. 4. Make effective use the Java Software Development Kit Application Programming Interfaces.	1. Can design, construct, test and document a Java program, given a problem specification of limited complexity.

9a. The Master Module 2 and Module 2.2 are similar.

Answer / *Criteria*	*Strongly Agree*	*Agree*	*Neutral*	*Disagree*	*Strongly Disagree*
Keywords					
Learning Outcomes					
Overall					

9 b. Please provide your level of confidence in the answer to question 9a

Low	*Moderate*	*High*